UNCLASSIFIED

AD NUMBER

AD812885

NEW LIMITATION CHANGE

TO

Approved for public release, distribution unlimited

FROM

Distribution authorized to U.S. Gov't. agencies and their contractors; Critical Technology; MAR 1967. Other requests shall be referred to Rome Air Development Center, Attn: EMIIF, Griffis AFB, NY 13440.

AUTHORITY

RADC, USAF ltr, 17 Sep 1971

THIS PAGE IS UNCLASSIFIED

AD812885

RADC-TR-66-696
Final Report

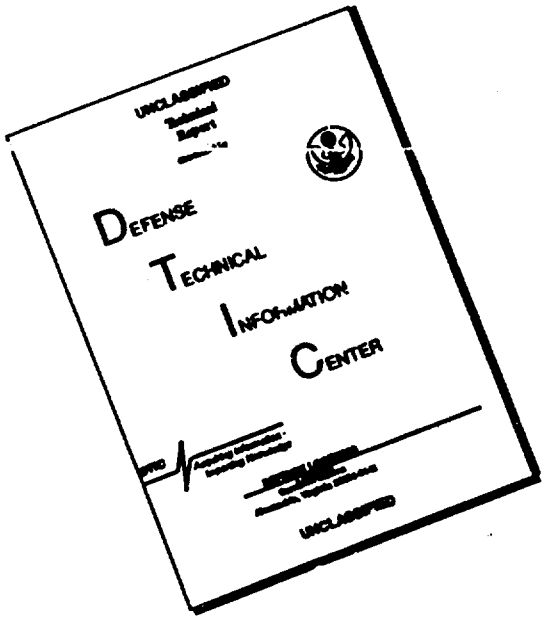
COMPUTER PROGRAM FOR AUTOMATIC SPELLING CORRECTION

Joseph A. O'Brien
Itek Corporation

TECHNICAL REPORT NO. RADC-TR-66-696
March 1967

Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

DISCLAIMER NOTICE

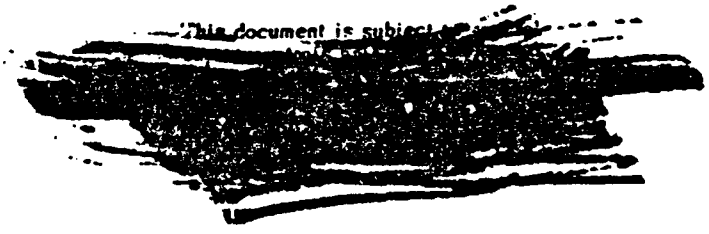


THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

COMPUTER PROGRAM FOR AUTOMATIC SPELLING CORRECTION

Joseph A. O'Brien

Itek Corporation



AFLC, GAFB, N.Y., 20 Apr 67-100

FOREWORD

This final report was prepared by Itek Corporation, Lexington, Massachusetts under Contract No. AF30(602)-3484, Project 4594. The report is identified by the contractor as No. 66-8428-1.

RADC Project Engineer was Louis Comito (EMIIF).

This document is not releasable to CFSTI because it contains information embargoed from release to Sino-Soviet Bloc countries by AFR 400-10, "Strategic Trade Control Program."

This report has been reviewed and is approved.

Approved: FRANK J. TOMAINI
Chief, Info Processing Branch
Intel & Info Processing Division

Approved: JAMES J. DIXEL, Colonel, USAF
Chief, Intel & Info Processing Division

FOR THE COMMANDER:
IRVING J. GABELMAN
Chief, Advanced Studies Group

ABSTRACT

This technical documentary report, prepared under contract AF30(602)-3484, describes the logic and operation of the Spelling Correction program prepared by Itek Corporation for RADC. The program is written in RADCAP language to operate on a CDC 8090 computer under control of the RADC Experimental Computer Complex (ECC). The various routines which compose the Spelling Correction program are fully described. General flow charts of the program as well as detailed descriptions, flow charts, and operating instructions for each routine are included. This report provides sufficient information to enable the user to make maximum use of the program's capabilities.

CONTENTS

1. Introduction 1-1

2. Summary 2-1

3. System Description 3-1

4. Program Routines 4-1

4.1 PEELER Routine 4-1

4.2 SORT (SOALNO) Routine 4-8

4.3 MERGE Routine 4-9

4.4 SHORTWORD (SHRTWD) Routine 4-21

4.5 LOAD DISK (LODDSK) Routine 4-26

4.6 SHIFT REGISTER COMPARATOR (SRC) Routine 4-32

4.7 RE-SORT Routine 4-39

4.8 STATISTICS I (WORST) Routine 4-51

4.9 STATISTICS II (PONCW) Routine 4-52

4.10 DISPLAY Routine 4-58

4.11 MANUAL DATA EDIT (MDE) Routine 4-62

4.12 CODE CONVERSION (CONVRT) Routine 4-67

4.13 GENERATION OF SIMULATED DATA Program 4-67

5. Operating Procedures 5-1

5.1 PEELER Operation 5-1

5.2 SORT Operation 5-1

5.3 MERGE Operation 5-2

5.4 SHORTWORD Operation 5-2

5.5 LOAD DISK Operation 5-3

5.6 SHIFT REGISTER COMPARATOR Operation 5-3

5.7 RE-SORT Operation 5-4

5.8 STATISTICS I Operation 5-4

5.9 STATISTICS II Operation 5-5

5.10 DISPLAY and MANUAL DATA EDIT Operation 5-5

5.11 CODE CONVERSION Operation 5-6

5.12 GENDAT Operation 5-6

Appendix A. Description of an "Item" A-1

Appendix B. Batching Scheme Used in Spelling Correction Program B-1

Appendix C. End-of-File Blocks C-1

Appendix D. Sequence for Setting Shift Register Comparator Conditions . D-1

Appendix E. Examples of SHIFT REGISTER COMPARATOR Routine Flag Bits . . E-1

Appendix F. Examples of Output of STATISTICS I Routine F-1

FIGURES

3-1 Simplified block diagram of Spelling Correction program 3-2

3-2 System flow diagram for Spelling Correction program 3-3

4-1 Flow diagram for PEELER routine 4-2

4-2 Flow diagram for SORT routine 4-4

4-3 Flow diagram for MERGE routine 4-11

4-4 Flow diagram for SHORTWORD routine 4-22

4-5 Flow diagram for LOAD DISK routine 4-27

4-6 Flow diagram for SHIFT REGISTER COMPARATOR routine 4-33

4-7 Flow diagram for RE-SORT routine 4-41

4-8 Flow diagram for STATISTICS I routine 4-53

4-9 Flow diagram for STATISTICS II routine 4-56

4-10 Flow diagram for DISPLAY routine 4-59

4-11 Flow diagram for MANUAL DATA EDIT routine 4-64

4-12 Examples of 408-A Datacom display when using MANUAL DATA EDIT routine 4-68

4-13 Flow diagram for CODE CONVERSION routine 4-69

4-14 Flow diagram for GENDAT program 4-72

1. INTRODUCTION

The rapid development of computer technology justifies the use of computers in varied and unique applications. The Spelling Correction program is a prime example of a computer application which now appears both technologically and economically feasible.

The Spelling Correction program is a prototype design that will function as a basic model for developing more sophisticated spelling correcting techniques in the future. The method of correcting input text is by dictionary lookup. The lookup process utilizes a standard disk storage device to retain the working dictionary and a special shift register comparator to perform the matching function. The correction logic and data handling functions are performed by the programmed routines.

The Spelling Correction program was designed to operate in conjunction with the RADC Experimental Computer Complex (ECC). The ECC is a diverse assemblage of computers and peripheral devices controlled by an operating system called the Executive Control Program (ECP). The objective of the ECC and the ECP systems is to provide the user with a maximum of capability for a minimum of effort. To accomplish these objectives, extensive software systems were written to provide the user with automatic program segmentation, seemingly unlimited storage capacity, and a complete file management service. One of the software systems (i.e., assembly programs) operating under the ECP and providing the above capabilities is called RADCAP. This is the assembly language specified by RADC for the Spelling Correction program.

The hardware configuration of the ECC system includes a CDC 8090 computer for processing RADCAP programs. (The 8090 is basically a CDC 160A computer except for its physical appearance.) The peripheral equipment of the ECC complex used by the Spelling Correction program includes an IBM Selectric typewriter for system communication, RCA and Univac servo magnetic tape units for program and intermediate data storage, a paper tape reader and punch, an LFE BD-500 disk unit, a 408-A Datacom display, and a Philco shift register comparator.

Since the ECC is an experimental system, the hardware and software configuration did undergo numerous changes during the contract period. In some instances these changes had a direct effect on the operation and development of the Spelling Correction program. In this sense, the Spelling Correction project assumed more of an experimental nature than was originally planned.

The contract period allotted time for the various routines of the Spelling Correction program to be tested and debugged. However, the full capabilities and limitations of the program can be realized only after extensive production runs are performed.

2. SUMMARY

The Spelling Correction program was conceived by RADC as an automatic method of correcting errors in textual information. The basic design of the correction process and the hardware-software configuration for its operation were specified by RADC. Owing to the experimental nature of the operating systems, some revisions to the original specifications were required during the course of this contract.

The initial input to the Spelling Correction program is the digitized output of documents processed by an optical print reader. This information is then converted to an appropriate code for the specified hardware. Various sorting, merging, and word identification routines prepare the information for the lookup process, which is handled by dictionary lookup techniques. The correction process is geared to a shift register comparator device attached to the operating computer. The converted text is then reconstructed, with the corrected words replacing those found in error.

In the initial phase of the project, Itek conducted a statistical analysis to determine the frequency and type of errors in the documents to be processed. This information, supplied under separate cover, provided a basis for early testing of the program.

The Spelling Correction program itself generates additional statistics on errors that might permit further refinements of the technique. To prove the feasibility and efficiency of this method of error correction, operational tests should be conducted using the program on actual data in a production environment.

3. SYSTEM DESCRIPTION

The objective of the Spelling Correction program is to correct certain errors in the spelling of words contained in text processed by a print reading machine. The program is designed to operate in conjunction with a special comparator device (the Philco shift register comparator) controlled by a CDC 160/160A computer. The primary errors to be corrected by the configuration are those which existed in the text words prior to print reading because of human and mechanical factors other than those associated with the print reader itself.

The method of correcting input text is by comparison with stored dictionary data from a Laboratory for Electronics BD-500 magnetic disk file storage unit, which also operates under control of the CDC 160/160A computer. Fig. 3-1 is a generalized block diagram of the system.

The Spelling Correction program is presently composed of 12 routines and a dictionary of the 5,000 most common English words. Two of the routines gather statistical information which is intended to serve as feedback to suggest improvements or refinements to the correction process. The remaining 10 routines are actually involved in data handling and/or controlling the correction processes.

The Spelling Correction program is written in 160A assembly language called RADCAP-Stage 1. The various routines are segmented into individually operating sections of 512 (12-bit) computer words. All input-output data handling is accomplished using blocks of the same size. All internal processing in the Spelling Correction program is done in Universal code. This provides complete compatibility between the subroutines and the objectives of the Executive Control Program (ECP) under which they operate.

RADCAP-Stage 1 was chosen over RADCAP-Stage 2 as the working language because of its far greater assembly rate. The Stage 1 routines are completely compatible with Stage 2. Thus, the routines can be assembled under Stage 2, without modification, to take advantage of the more efficient object language produced by Stage 2.

The general flow of the Spelling Correction routines (see Fig. 3-2) is essentially one straight pass through the 12 routines except for the interaction which takes place between the MANUAL DATA EDIT routine and the DISPLAY routine. Each routine is designed to accept the output of the previously executed routine as input. All intermediate information is passed using magnetic

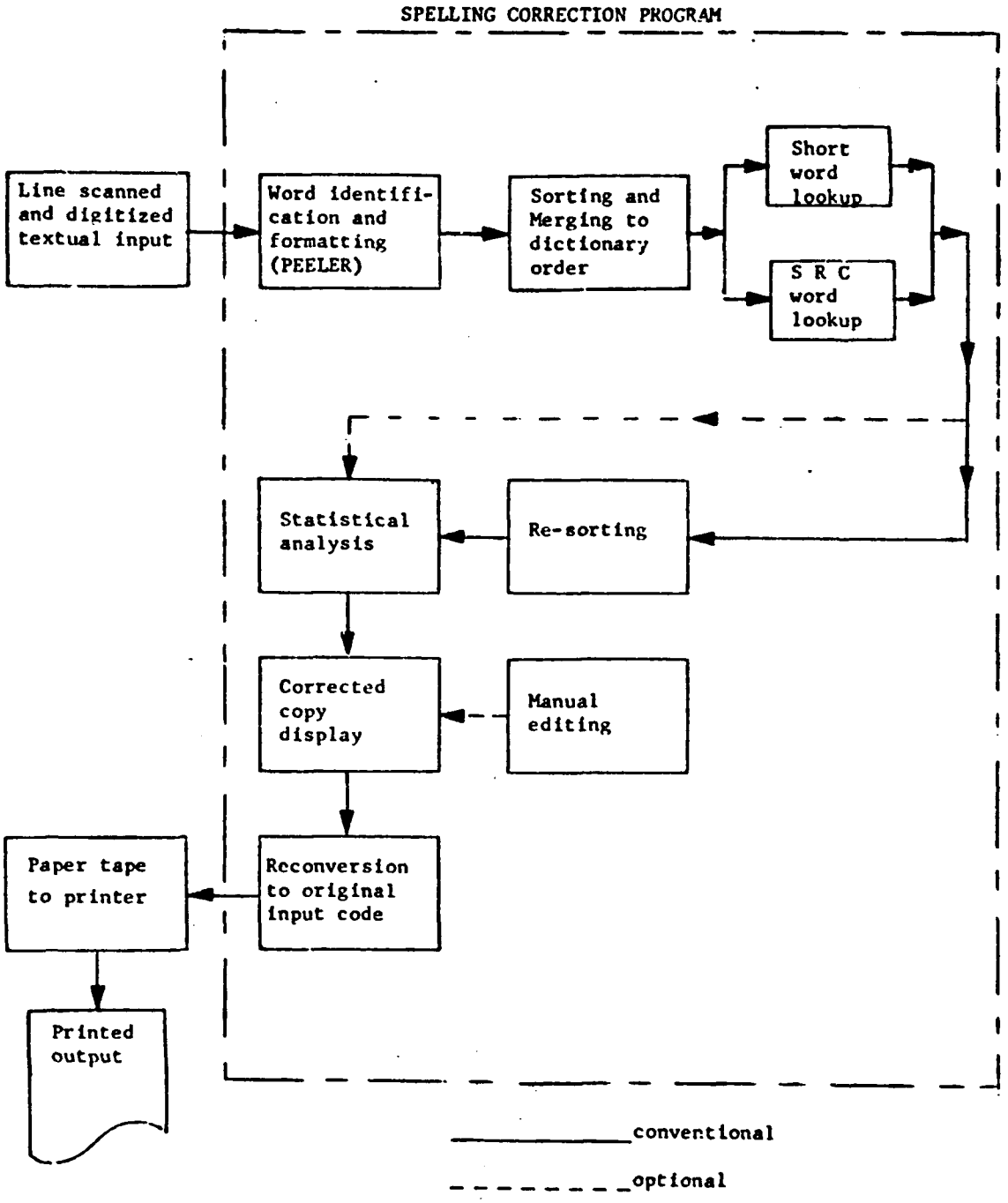


Fig. 3 - 1 Simplified block diagram of Spelling Correction Program

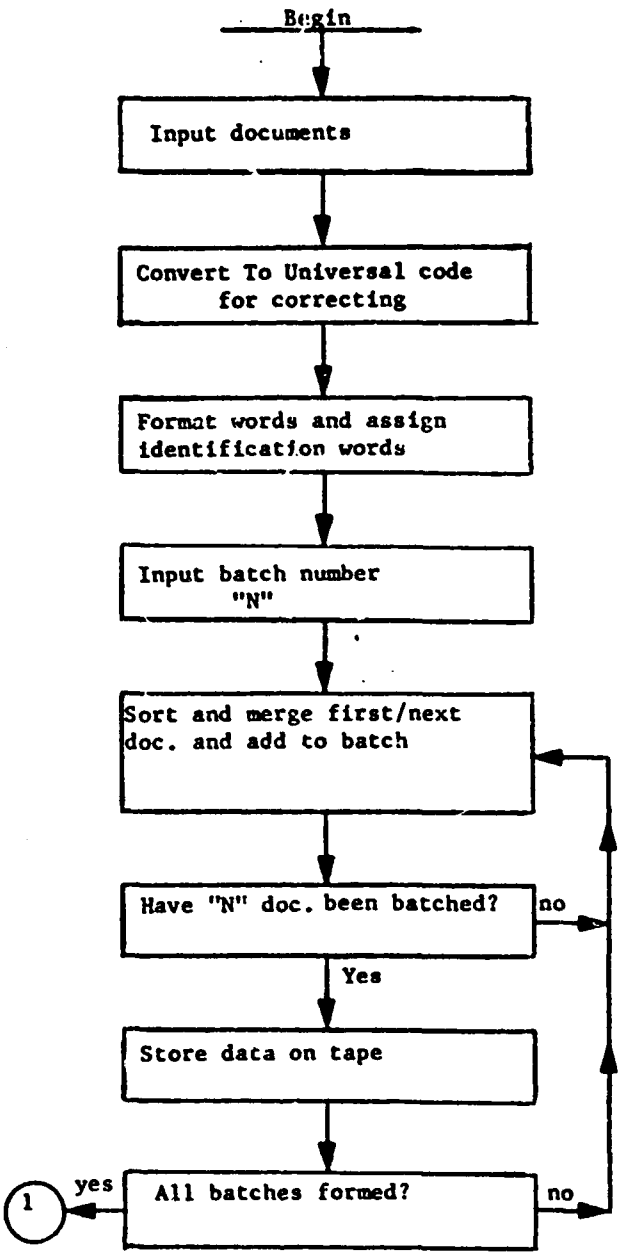


Fig. 3 - 2 System flow diagram for Spelling Correction Program

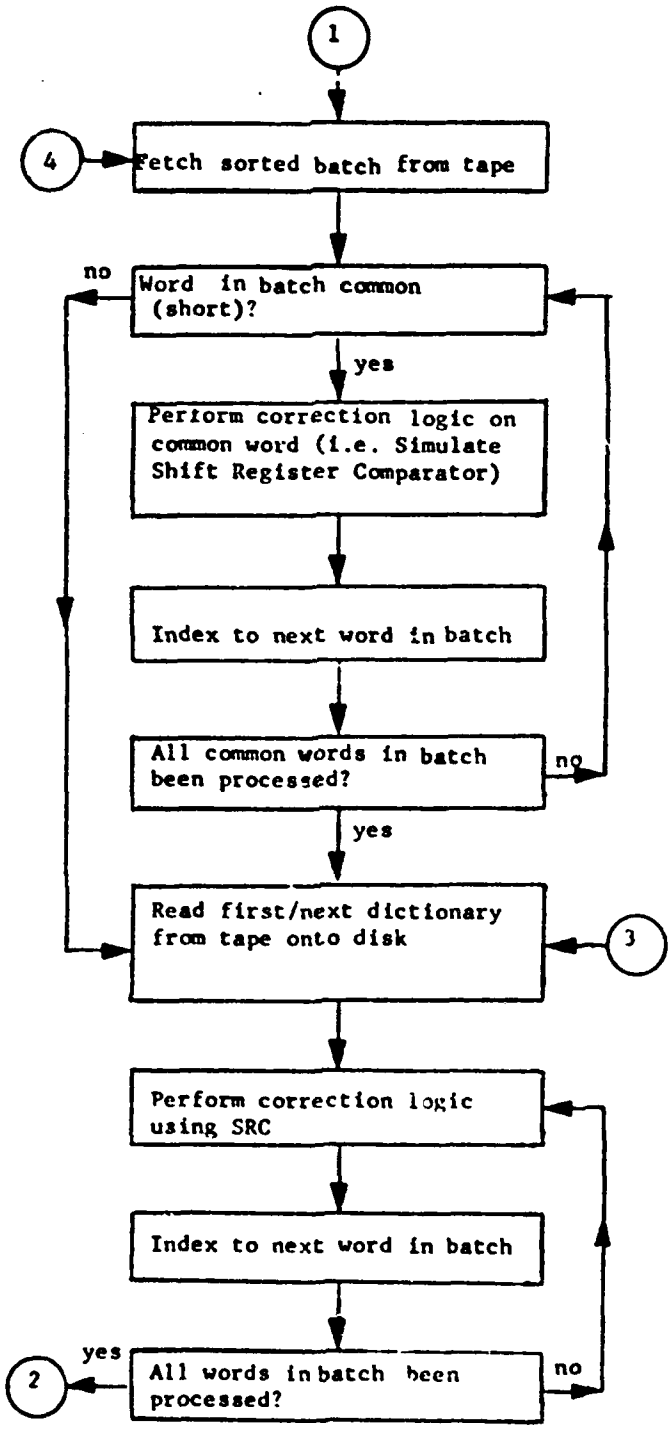

1
4
Fetch sorted batch from tape
no
Word in batch common (short)?
yes
Perform correction logic on common word (i.e. Simulate Shift Register Comparator)
Index to next word in batch
All common words in batch been processed?
no
yes
Read first/next dictionary from tape onto disk
3
Perform correction logic using SRC
Index to next word in batch
yes
2
All words in batch been processed?
no

Fig. 3 - 2 (cont'd)

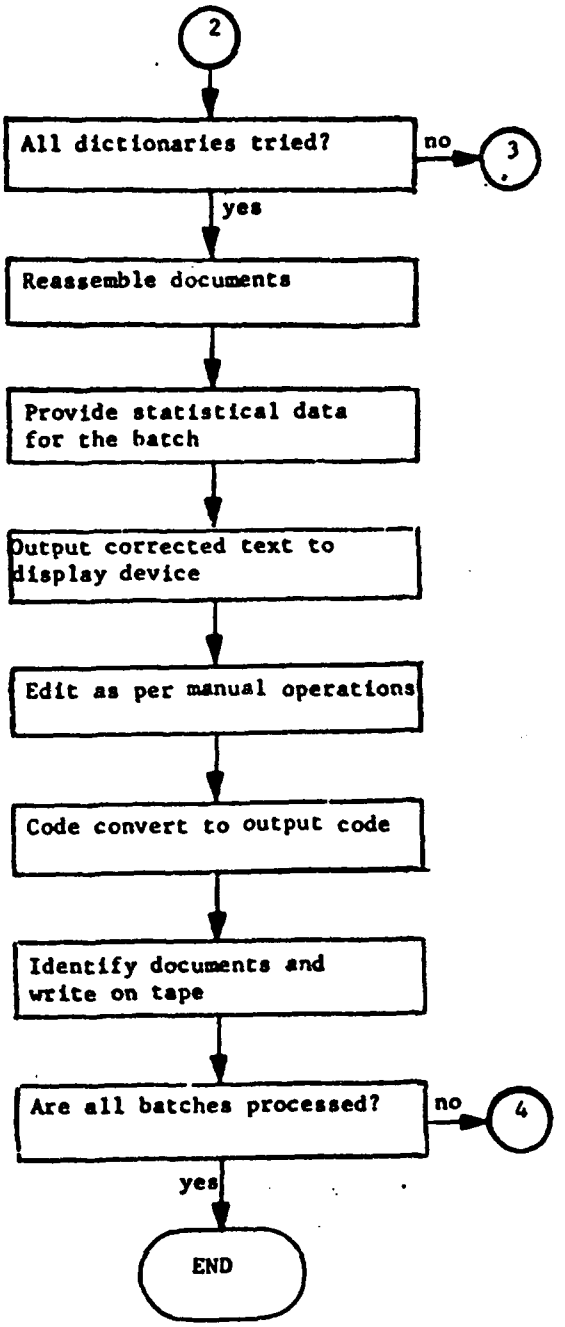

2
All dictionaries tried?
no
3
yes
Reassemble documents
Provide statistical data for the batch
Output corrected text to display device
Edit as per manual operations
Code convert to output code
Identify documents and write on tape
Are all batches processed?
no
4
yes
END

Fig. 3 - 2 (cont'd)

tape. A more detailed description of the function of each routine is available from the individual writeups and flow charts in Section 4.

The separation of the initial stream of characters from the Print Reader Conversion program into data words for the correction process is accomplished by the PEELER routine. Each data word is then appended with various descriptive words which are necessary for later processing. These descriptive words include information relative to the position of the data word in the original input stream, data word delimiters, etc. A data word and its associated descriptive words is called an "item" of information. (See Appendix A for an example of an item.) The items are then packed into blocks of 512 computer words (12-bit words) for input to the SORT routine.

The SORT routine sorts the items within each block in ascending order of character length. In addition, the blocks of data words are combined, or "batched," according to information supplied by an operator. The MERGE routine merges the blocks of data words which compose a batch. This is performed using the sorting logic of the SORT routine. (See Appendix B for more detailed information regarding batching.)

The correction techniques applied to the data words are performed in the SHIFT REGISTER COMPARATOR (SRC) and SHORTWORD routines. The order in which these two routines are executed is immaterial.

The SRC routine utilizes the shift register comparator to correct the data words. As stated previously, the correction is accomplished using dictionary lookup methods. The dictionary entries are on paper tape in the Universal character code delimited by the appropriate beginning and end words required by the shift register comparator. The order of the dictionary entries on paper tape is identical to the order of the data words in each batch produced by the SORT and MERGE routines.

Before the SRC routine can initiate the lookup techniques, the dictionary must be loaded from paper tape onto the LFE BD-500 disk. If the dictionary tape is too long to fit on the disk in a single load, it must be separated by single end-of-file blocks. The last dictionary entry should be followed by a 177_8 code following the usual 175_8 end-of-word code.

The primary function of the SRC routine is to control the use of the shift register comparator. The comparator is capable of operating in more than one mode, but with this routine operation is restricted to a single mode. Words to be looked up are loaded into one register of the comparator and the dictionary entries on the disk are cycled through a second comparator register. When the two registers in the comparator are matched, within a program set tolerance level, the data word is considered corrected. The area of the disk used in the lookup search and the tolerance level (i.e., threshold level) are controlled by the SRC routine. This routine is essentially the "heart" of the Spelling Correction program, since the correction logic and techniques are contained here.

Due to physical limitations of the comparator, it is capable of handling only data words from 4 to 18 characters in length. Data words containing more than 18 characters are very rare and are ignored by the Spelling Correction

program. Words containing less than four characters are corrected in the SHORTWORD routine. This routine simulates the operation of the shift register comparator using programmed instructions. The logic, threshold limits, program flags, etc., are identical to the SRC routine. Thus, following the operation of the SRC and SHORTWORD routines, the automatic correction of data words is completed.

The RE-SORT routine resorts the sorted and merged data words to the original order of the print reader output stream. This is accomplished using the document number and item position number in the descriptive words. After the documents are back in their original order, all descriptive words are removed by the DISPLAY routine. Then the document is displayed on the 408-A Datacom with a standard indentation of nine spaces, which is used by the edit portion of the routine, and an additional indentation of five spaces whenever a data line exceeds the scope line. When the data has been displayed, it is again possible to correct misspelled words. However, here the corrections are made manually by an operator at the display console. If editing is to be performed on a complete line, or lines, the operator may utilize the various subroutines contained in the MANUAL DATA EDIT routine. At present there are three subroutines for this purpose, but the program structure is designed so that additional subroutines can easily be included.

Once the data has been manually corrected, the text is converted back to the original print reader code. This is done by the CODE CONVERSION routine in order to obtain a hard copy of the corrected text without having to combine codes to represent special characters.

Although the correction process is completed at this point, there still remain two other routines; both are statistical in nature and both use the output of the SRC routine.

The first statistical routine, WORST, is designed to produce a table showing the composition of the input text and the effectiveness of the spelling correction procedures. This is done by classifying each item by character length, presence or absence of a confusion character, confusion character and/or best guess, and whether the words are corrected or remain uncorrected. The words are also examined in order to obtain an average threshold level. Frequency counts are kept for each of the categories according to character length. (See Appendix F for a sample output of this routine.)

The second statistical routine, PONCW, is designed to give a printout of all alphabetic items that were not corrected automatically. The printout of each uncorrected word is preceded by its tolerance level. Within each item all confusion characters are denoted by an asterisk.

4. PROGRAM ROUTINES

4.1 PEELER ROUTINE

4.1.1 Purpose

The purpose of this routine is to define data words for the Spelling Correction routines and to append certain descriptive words to each data word.

4.1.2 Input

The input for this routine is a magnetic tape, outputted from RADC's Printer Code Conversion program, which contains the Universal codes substituted for original Print Reader codes.

4.1.3 Description

The PEELER routine (see Fig. 4-1) processes the data stream from the input tape in sequential order and logically determines how many characters constitute a data word. A data word may be a single blank space, a mark of punctuation, a completely alphabetic stream of characters, numbers, etc.

To each data word six descriptive words are added. They include character count, which states the number of characters in the data word; type code, which indicates the composition of the data word (e.g., pure alphabetic, numeric, alphabetic with confusion characters, alphabetic with best guess); document number; position of data word within a document; and beginning and end words, 174_8 and 175_8, respectively, necessary for the shift register comparator.

4.1.4 Output

A data word and its six descriptive words, referred to as an "item," are packed into 512-word blocks and outputted. (See Appendix A for examples of an item.) Unused locations at the end of a block (only complete items are stored in a block) are set to zero. When an end-of-document mark is encountered, the buffer is emptied onto the output tape followed by a single end-of-file block. The final block of the last document is followed by two consecutive end-of-file blocks on the output tape.

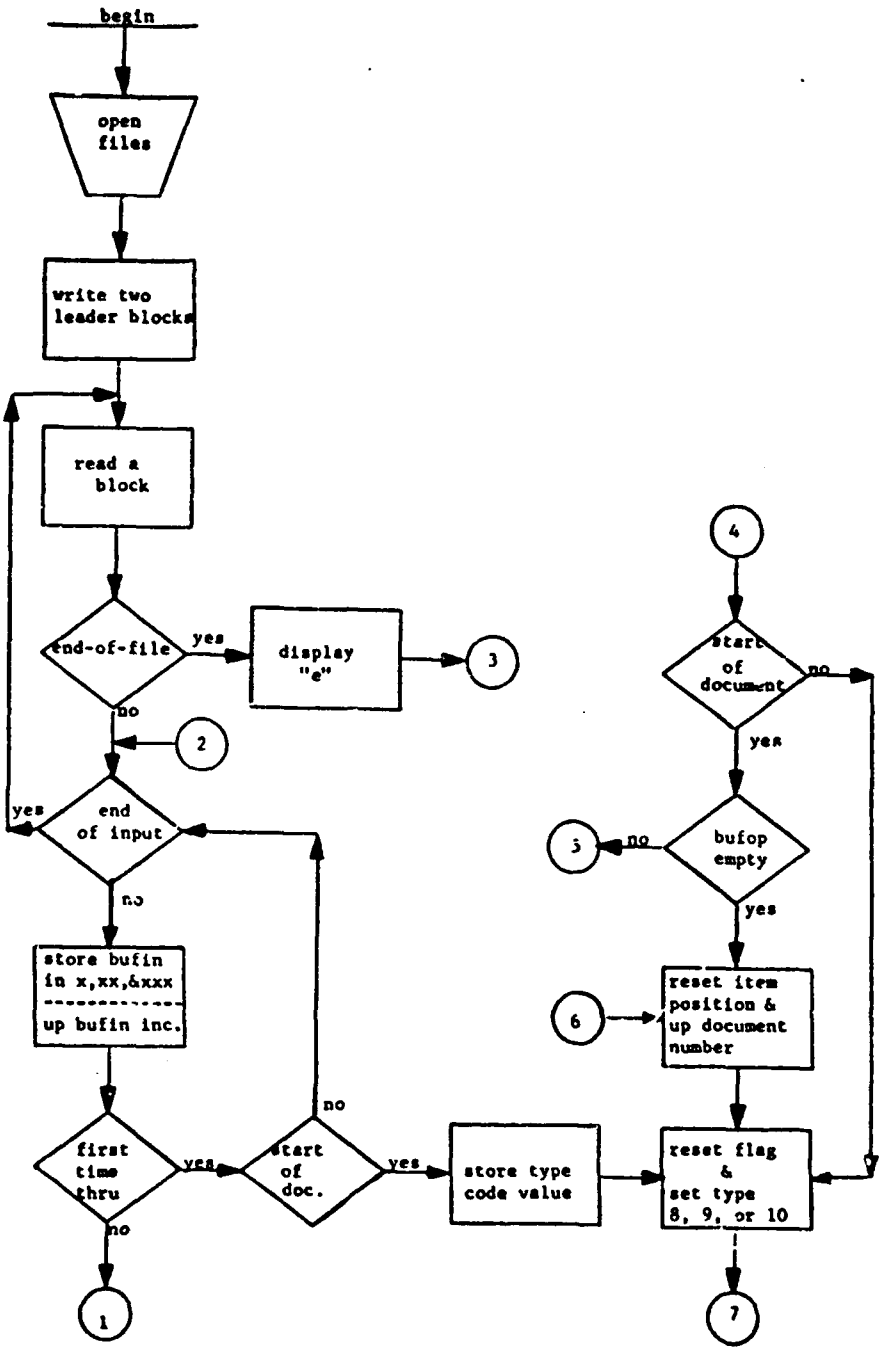


Fig. 4 - 1 Flow diagram for Peeler routine

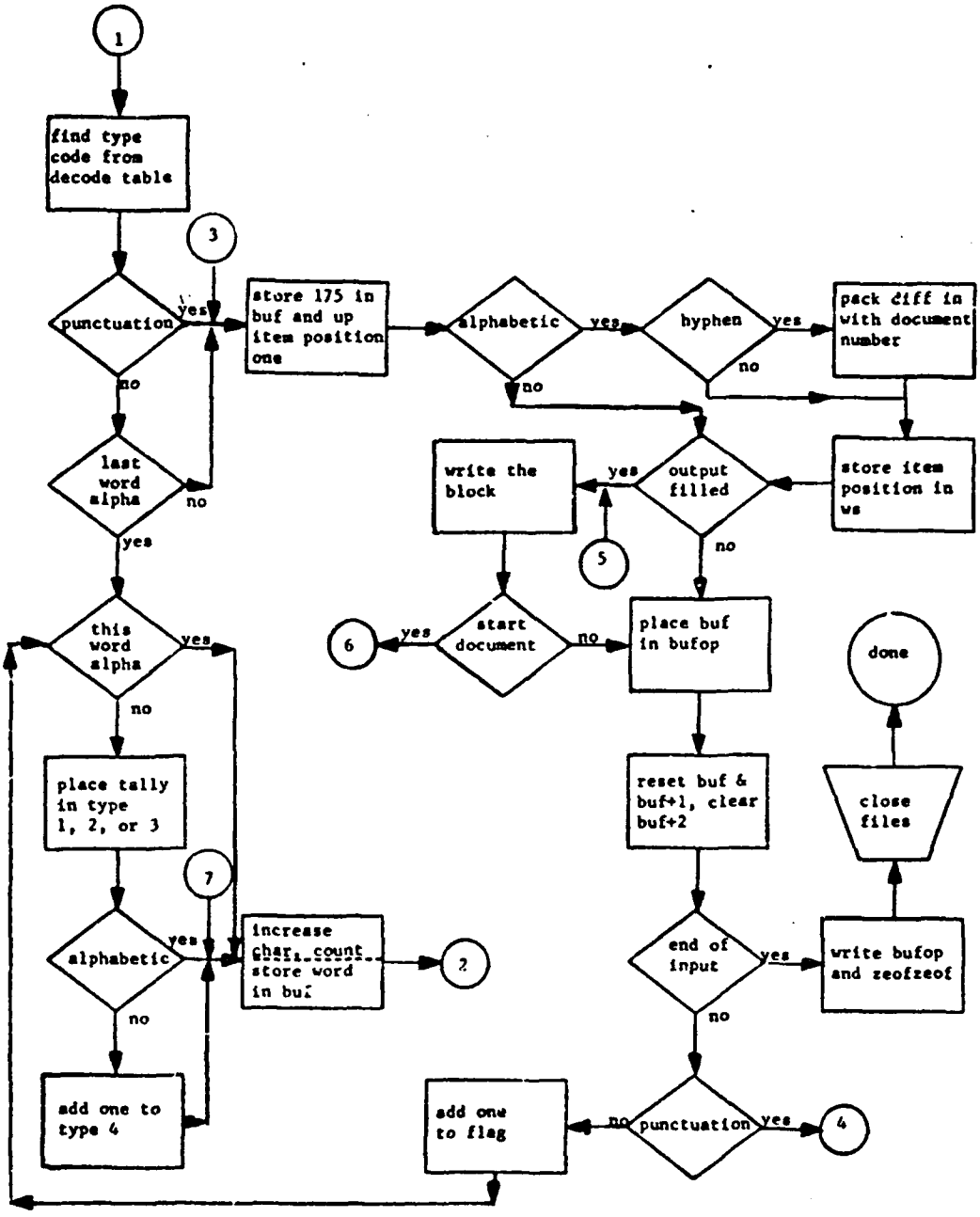

1
find type code from decode table
3
punctuation
yes
store 175 in buf and up item position one
alphabetic
yes
hyphen
yes
pack diff in with document number
no
no
no
last word alpha
no
yes
write the block
yes
output filled
store item position in ws
5
no
this word alpha
yes
6
yes
start document
no
place buf in bufop
done
no
place tally in type 1, 2, or 3
reset buf & buf+1, clear buf+2
close files
7
alphabetic
yes
increase char_count store word in buf
2
end of input
yes
write bufop and zeofzeof
no
no
add one to type 4
add one to flag
no
punctuation
yes
4

Fig. 4 - 1 (cont'd)

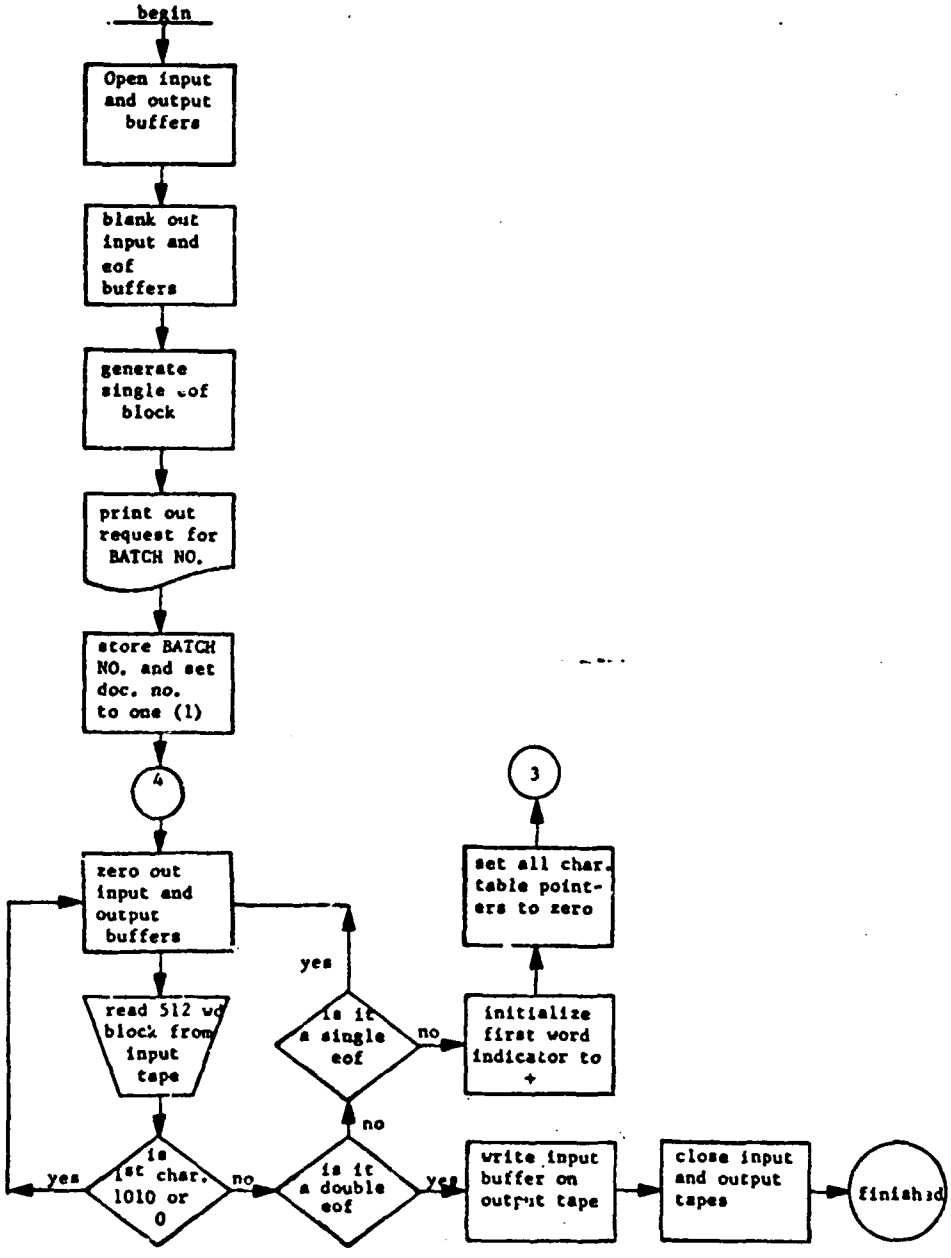


Fig. 4 - 2 Flow diagram for Sort routine

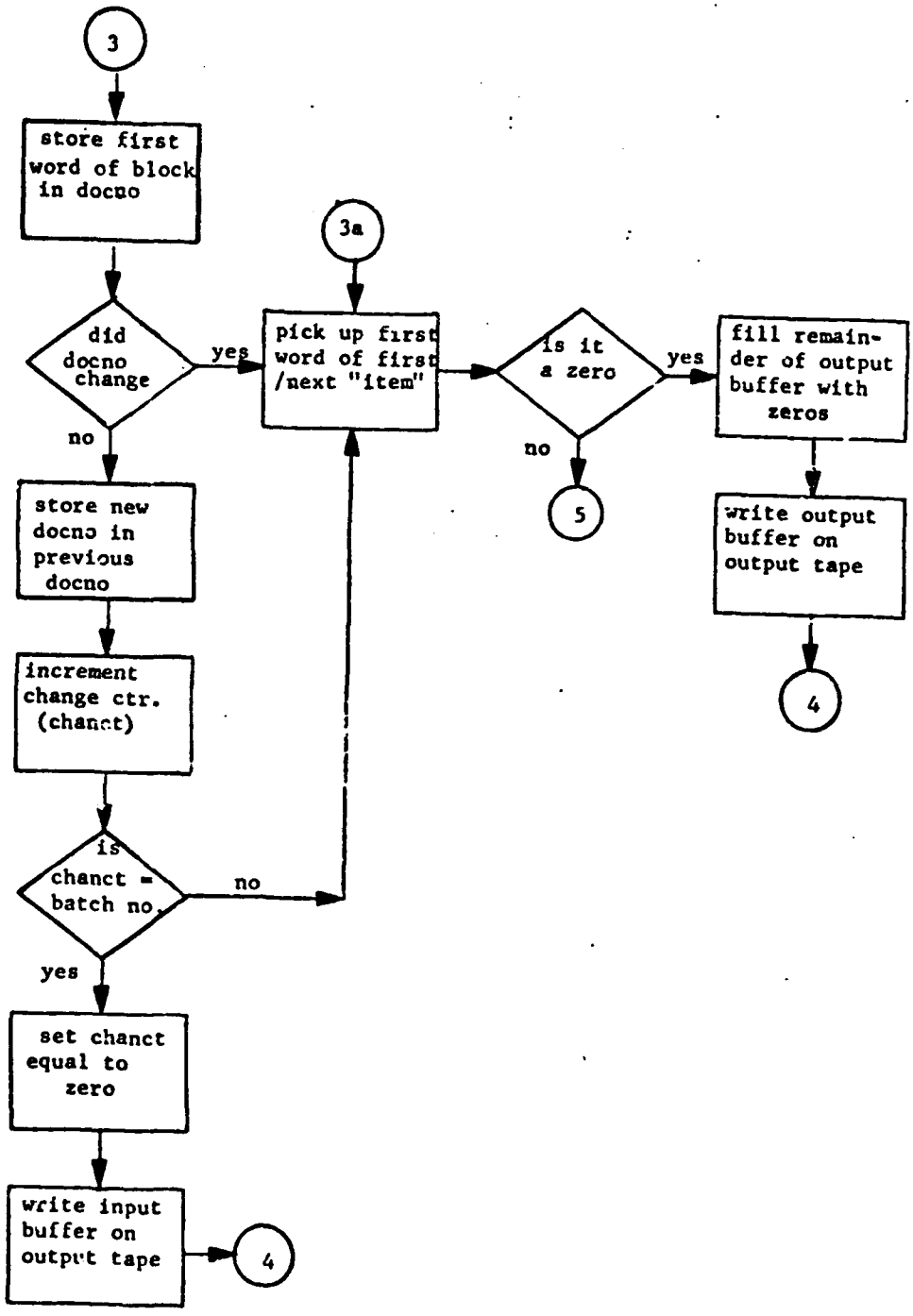

3
store first word of block in docno
3a
did docno change
yes
pick up first word of first /next "item"
is it a zero
yes
fill remainder of output buffer with zeros
no
store new docno in previous docno
no
5
write output buffer on output tape
4
increment change ctr. (chanct)
is chanct = batch no.
no
yes
set chanct equal to zero
write input buffer on output tape
4

Fig. 4 - 2 (cont'd)

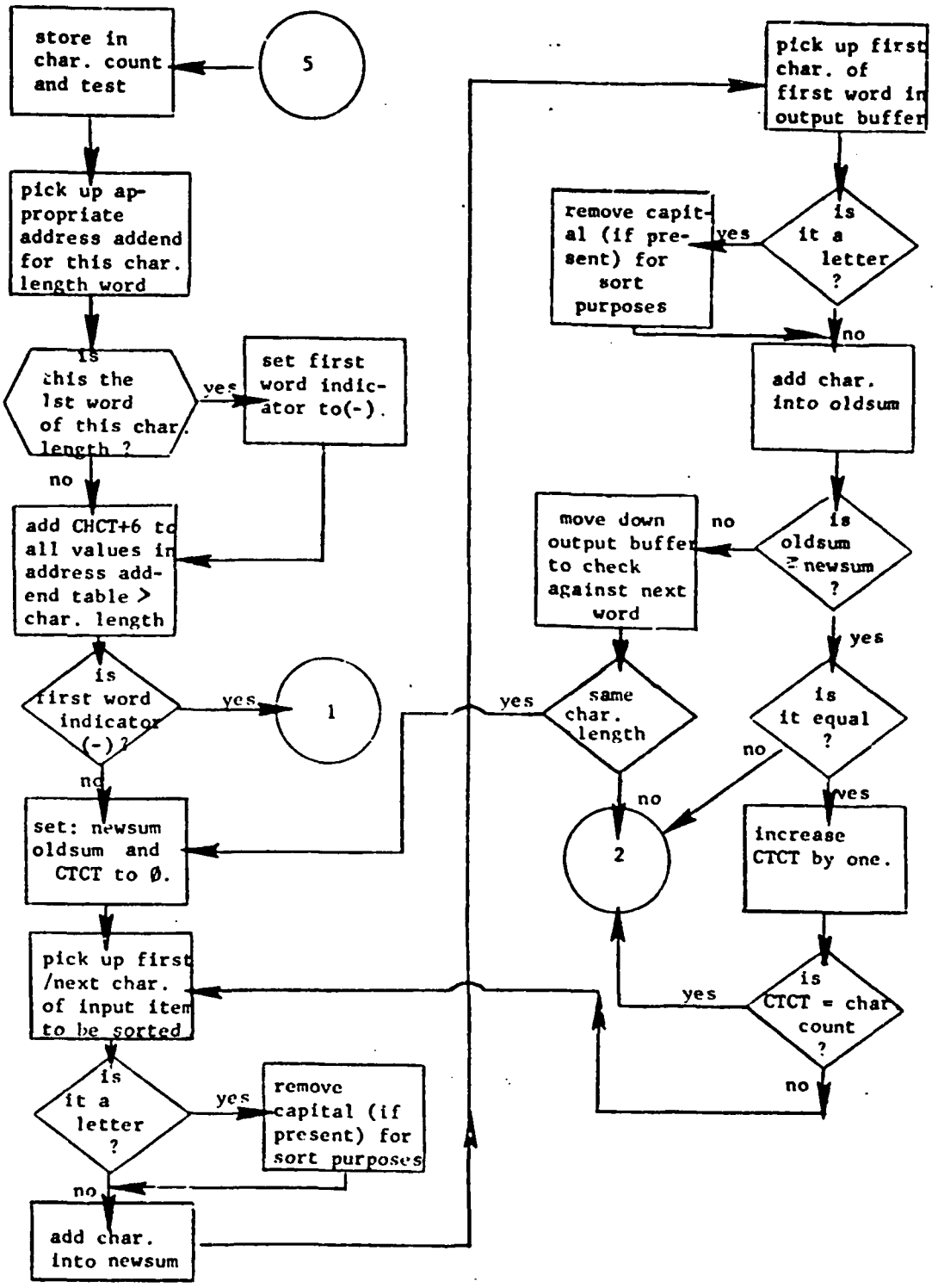

store in char. count and test
5
pick up appropriate address addend for this char. length word
is this the 1st word of this char. length ?
yes
set first word indicator to(-).
no
add CHCT+6 to all values in address addend table > char. length
is first word indicator (-) ?
yes
1
no
set: newsum oldsum and CTCT to Ø.
pick up first /next char. of input item to be sorted
is it a letter ?
yes
remove capital (if present) for sort purposes
no
add char. into newsum
pick up first char. of first word in output buffer
is it a letter ?
yes
remove capital (if present) for sort purposes
no
add char. into oldsum
is oldsum ≥ newsum ?
no
move down output buffer to check against next word
yes
same char. length
yes
no
2
is it equal ?
no
yes
increase CTCT by one.
is CTCT = char count ?
yes
no

Fig. 4 - 2 (cont'd)

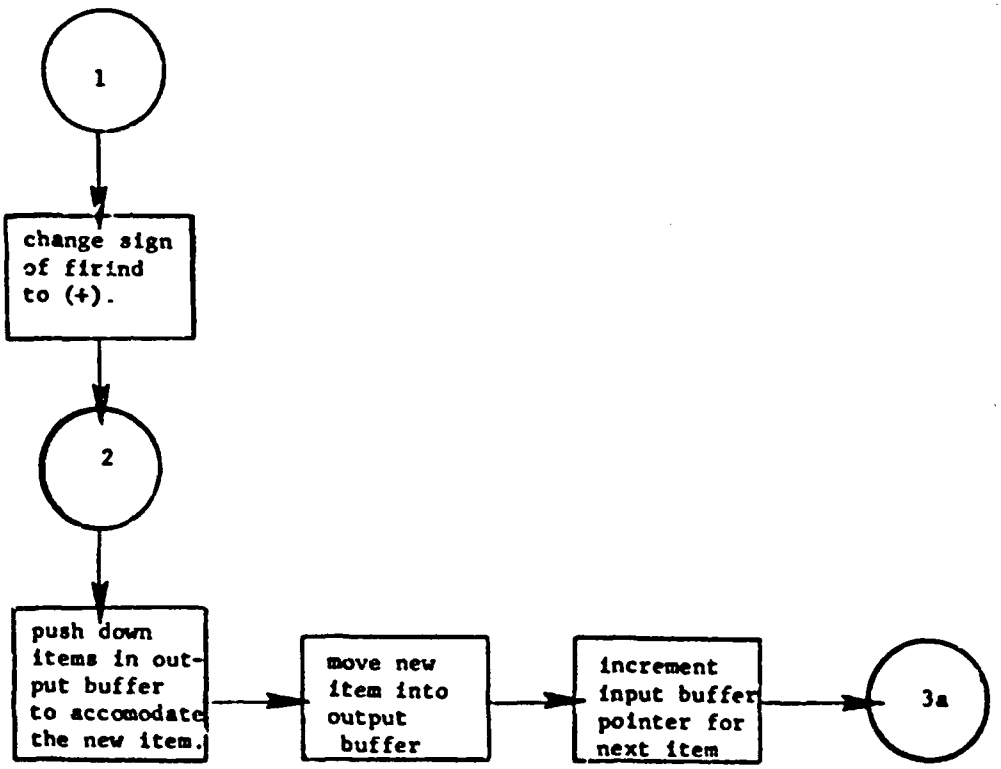

1
change sign of firind to (+).
2
push down items in output buffer to accomodate the new item.
move new item into output buffer
increment input buffer pointer for next item
3a

Fig. 4 - 2 (cont'd)

4.2 SORT (SOALNO) ROUTINE

4.2.1 Purpose

The purpose of this routine is to perform an internal sort on successive blocks of data which contain a variable number of items. The sort (see Fig. 4-2) is performed first according to character length and then alphanumerically within each character length.

4.2.2 Input

The input is a magnetic tape containing blocks of data consisting of 512_{10} words. Each block contains a variable number of items, where each item is of length L, and 7≤L≤96. The first word in a block must be a document number and the second word should contain the number of the last item in the block plus one. The items are packed in the remaining words of a block and must be complete. Unused words in a block should be equal to zero.

The last block of the input data should be followed by a double end-of-file block (see Appendix C for a description of an end-of-file block).

4.2.3 Function

Blocks of data are read sequentially from the input tape, one at a time, into core memory. Items of information are then taken from the input buffer and stored in an output buffer to obtain alphanumeric order within each character length. Items of shorter character length are stored at the top of the input buffer.

The alphanumeric sort within a fixed character length is determined by the Universal codes. Thus, numeric codes precede alphabetic codes, which in turn precede punctuation and special characters. When a block has been completely sorted, it is then written onto the output tape and the next block on the input tape is "read in" and processed.

4.2.4 Method

The input and output buffers are cleared to zeros preceding each read in from the input tape. A block is considered to contain legitimate information so long as the first word of the block is nonzero and it is not an end-of-file block.

The technique used in sorting is a pushdown list. The first item is placed in the top of the output buffer. The next, and each preceding, item is then positioned in the output buffer with regard only to the items presently stored in the output buffer. Thus the input buffer is emptied from top to bottom, and the output buffer is filled from top to bottom without gaps.

This technique is accomplished by using a smaller buffer which contains "address pointers" to the output buffer to indicate the location of the first word of each character length. Each time a new block of data is read in, all the pointers are cleared to zero. The buffer containing the address pointers

must be updated each time the position of an item is determined in the output buffer. More specifically, each time an item of say N characters has been correctly positioned, the address pointers for all items of length greater than N must be updated.

With this method it is a simple matter to determine if one is examining the first occurrence of an item of a given character length. If the address pointer of this item and the address pointer of an item which contains one more character are the same, it is a first occurrence. When storing the first occurrence of an item of a given character length, one obviously bypasses the alphanumeric search.

When the program performs an alphanumeric search, small letters and capital letters are treated identically. However, when storing an item in its proper position in the output buffer, the original contents of the input buffer are transferred.

4.2.5 Batching

The magnetic tape which serves as input to the SORT routine consists of a series of 512-word data blocks containing data words from any number of separate documents, the final data block being followed by a double end-of-file block. To perform an efficient sort and merge operation, the SORT routine has the capability of batching a fixed number of documents.

When the SORT routine is first executed it types out a request to "key-in" the number of documents to be batched. With this information, the SORT routine separates the input data into the requested batch size for processing by the remaining routines. Each batch is followed by a single end-of-file block and the final batch by a double end-of-file block.

4.2.6 Output

The output consists of a magnetic tape separated into batches of data blocks as described above. Within each data block the data words are sorted by length and universally within each length. The first two words of each input data block have been removed; thus each output data block will contain two additional zero words at the end.

4.3 MERGE ROUTINE

4.3.1 Purpose

The purpose of the MERGE routine (see Fig. 4-3) is to produce a tape or tapes containing blocks (512_{10}) of data that are in ascending order of character length and alphanumerically sorted within each character length. This routine merges the sorted blocks of N documents, where N is limited only by the length of the magnetic tape.

4.3.2 Input

The input is a magnetic tape or tapes containing blocks (512_{10}) of sorted data. The data in each block has been sorted according to character length,

in ascending order, and alphanumerically within each character length. An end-of-file block (zeof in the first four words of a 512_{10} block) denotes the logical break between batches of documents to be processed. A double end-of-file block (zeofzeof in the first eight words of a 512_{10} block) denotes the end of the job.

4.3.3 Description

In the following explanation, the notation MT1 denotes the input magnetic tape, MT2 and MT3 denote the intermediate merged tapes, and MT4 denotes the final merged tape.

Three buffers of 512_{10} words and two buffers of 50_{10} words are used as memory storage and work areas. These areas are labeled SORTED (input block), MERGED (intermediate storage), MEROUT (output block), BUF1 (buffer for a sorted word), and BUF2 (buffer for a merged word). The last two buffers are used for the basic merging operation.

For initialization, a block of data is read from MT1 into SORTED and immediately written out onto MT2 in order to establish merged data. A block of data is then read from MT1 into SORTED and a block of data is read from MT2 into MERGED. A word from SORTED is transferred to BUF1 and a word from MERGED is transferred to BUF2. These two words are then compared for the lowest character count. If the character counts are equal, a check is made for lowest alphanumeric value. Depending on the outcome of these tests, one of the two words is transferred to MEROUT. The emptied buffer is refilled and the comparison process repeated until MEROUT is completely filled or the next word to be stored into MEROUT does not fit completely. MEROUT is then written out onto MT3 and the buffer is cleared.

When MERGED empties before SORTED, it is refilled from MT2. The empty-fill process of MERGED will continue until SORTED is emptied or an end-of-file block is encountered on MT2. If an end-of-file block is encountered on MT2 before SORTED is empty, then the rest of SORTED is transferred to MEROUT. MEROUT is then written onto tape MT3. A logical process then switches the input-output functions of MT2 and MT3, SORTED and MEROUT are filled again, and the process continues. If SORTED empties before MERGED, the rest of the present temporary input tape is read in and transferred for output.

When a single end-of-file block is encountered on MT1, the batch of documents just merged is transferred to MT4. This is done in order to save tape manipulation when starting to merge the next batch of documents. On encountering a double end-of-file block the same process takes place, but control is then passed to the next routine.

4.3.4 Output

The output is a magnetic tape containing N batches of J documents with each batch of J documents completely sorted by character length, in ascending order, and alphanumerically within each character length. On this tape neither N nor J has a limit. This tape is used as input to the SRC or SHORTWORD routine.

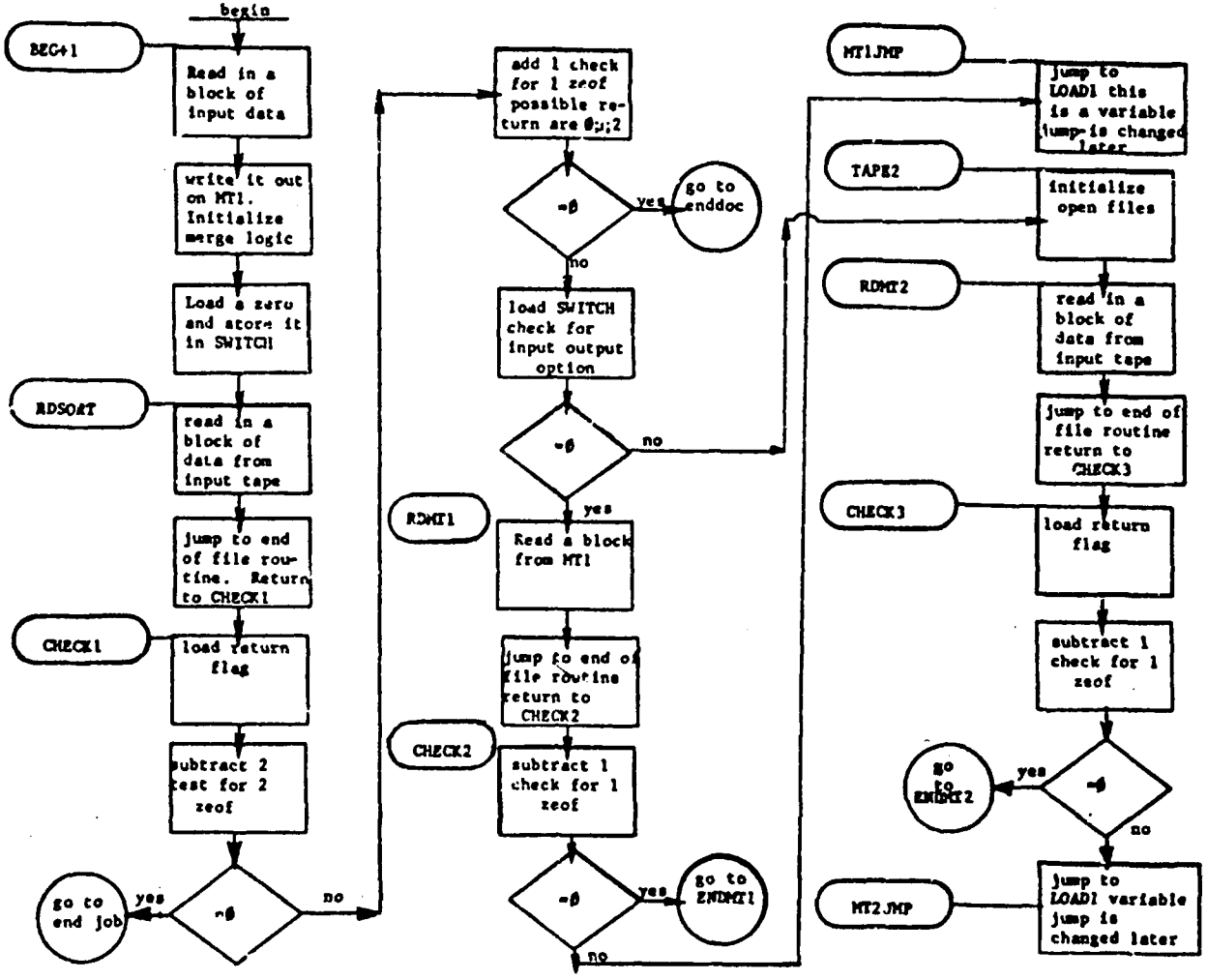


Fig. 4 - 3 Flow diagram for Merge routine

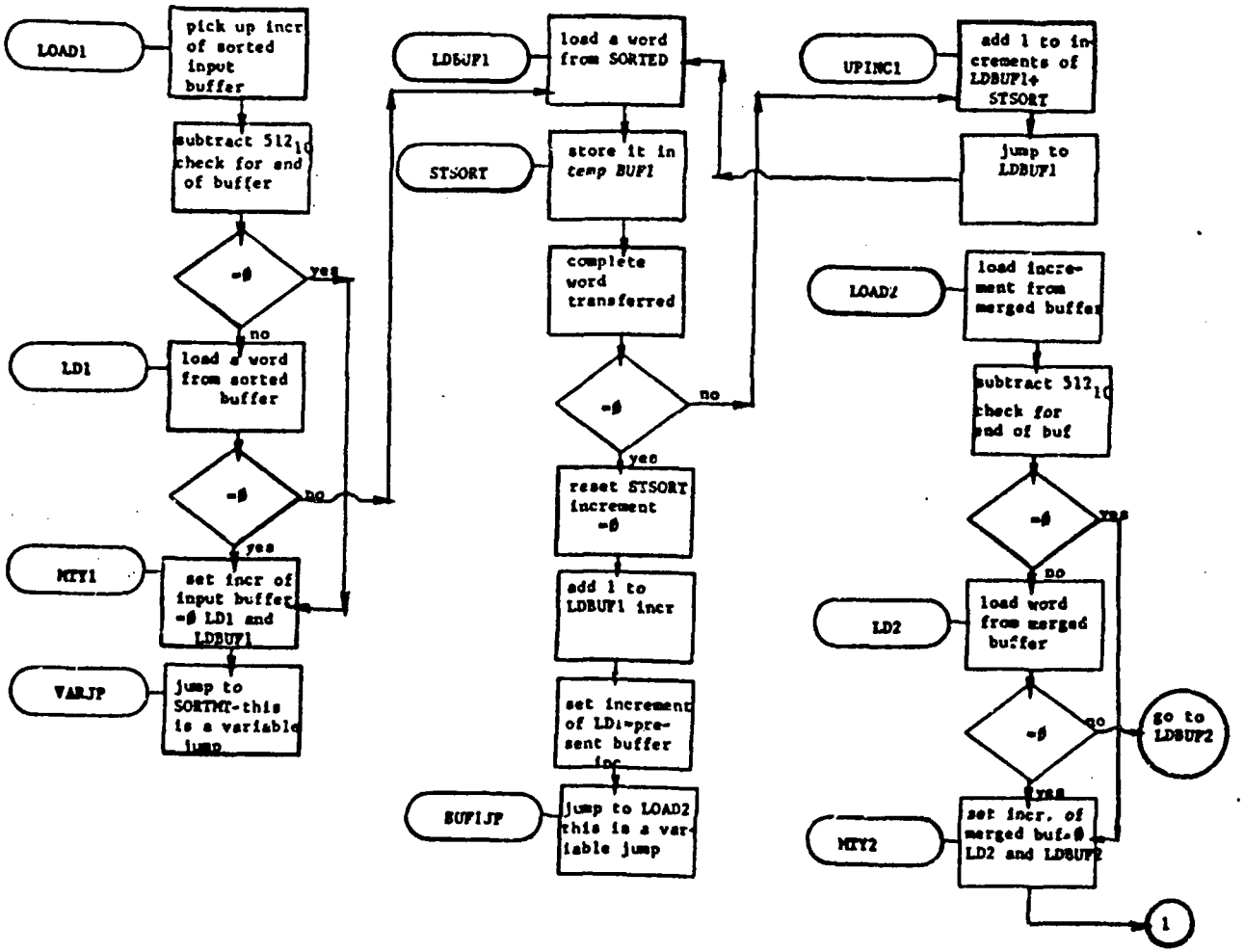

LOAD1
pick up incr of sorted input buffer
subtract 512₁₀ check for end of buffer
=ϕ
yes
no
LD1
load a word from sorted buffer
=ϕ
no
yes
MIY1
set incr of input buffer =ϕ LD1 and LDBUF1
VARJP
jump to SORTMT-this is a variable jump
LDBUF1
load a word from SORTED
STSORT
store it in temp BUF1
complete word transferred
=ϕ
no
yes
reset STSORT increment =ϕ
add 1 to LDBUF1 incr
set increment of LD1=present buffer inc
BUF1JP
jump to LOAD2 this is a variable jump
UPINC1
add 1 to increments of LDBUF1+ STSORT
jump to LDBUF1
LOAD2
load increment from merged buffer
subtract 512₁₀ check for end of buf
=ϕ
yes
no
LD2
load word from merged buffer
=ϕ
no
yes
go to LDBUF2
MIY2
set incr. of merged buf=ϕ LD2 and LDBUF2
1

Fig. 4 - 3 (cont'd)

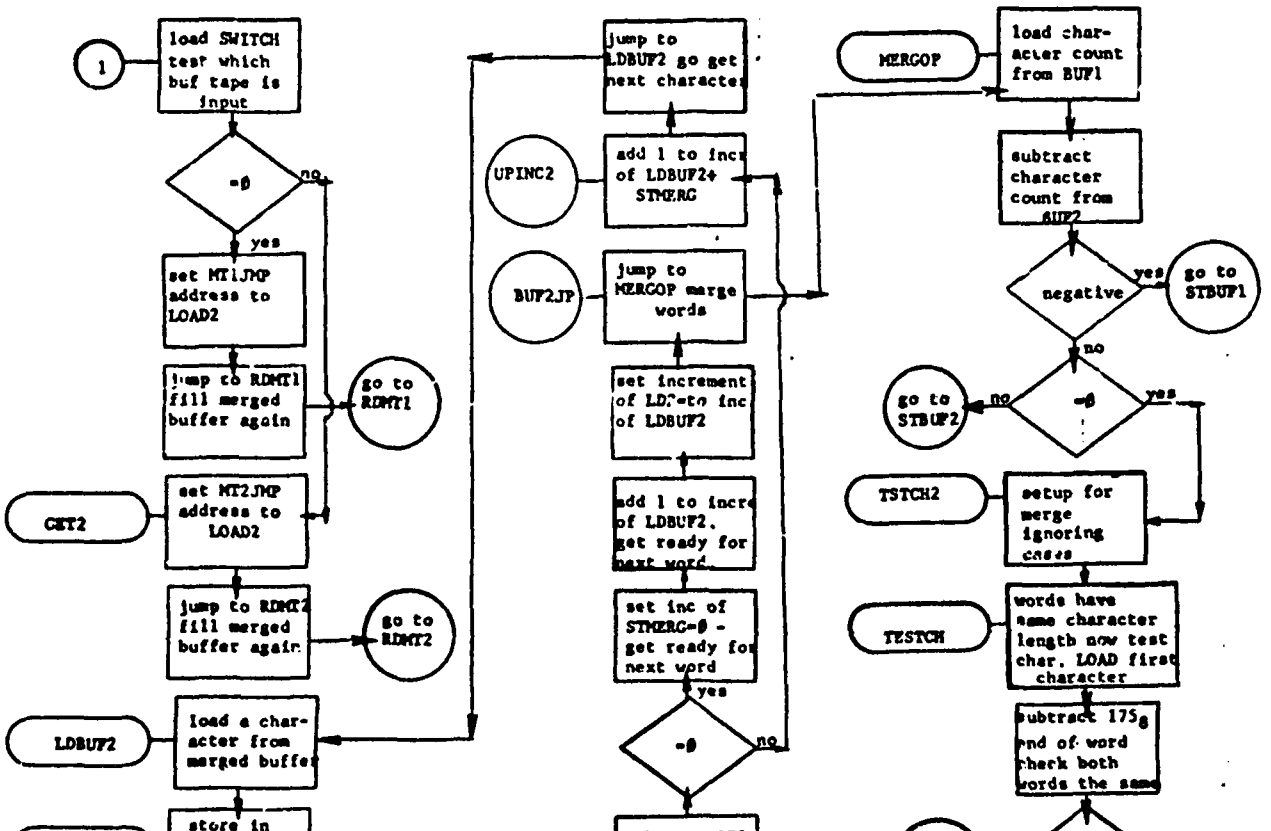

1
load SWITCH test which buf tape is input
=0
no
yes
set MT1JMP address to LOAD2
jump to RDMT1 fill merged buffer again
go to RDMT1
GET2
set MT2JMP address to LOAD2
jump to RDMT2 fill merged buffer again
go to RDMT2
LDBUF2
load a character from merged buffer
STMERG
store in temp buffer 2
jump to LDBUF2 go get next character
UPINC2
add 1 to incr of LDBUF2+ STMERG
BUF2JP
jump to MERGOP merge words
set increment of LDBUF2 to inc of LDBUF2
add 1 to incr of LDBUF2, get ready for next word
set inc of STMERG=0 - get ready for next word
=0
yes
no
subtract 175 end of word check
MERGOP
load character count from BUF1
subtract character count from BUF2
negative
yes
go to STBUF1
no
go to STBUF2
=0
yes
no
TSTCH2
setup for merge ignoring cases
TESTCH
words have same character length now test char. LOAD first character
subtract 175 end of word check both words the same
go to STBUF1
yes
=0
no
2

Fig. 4 - 3 (cont'd)

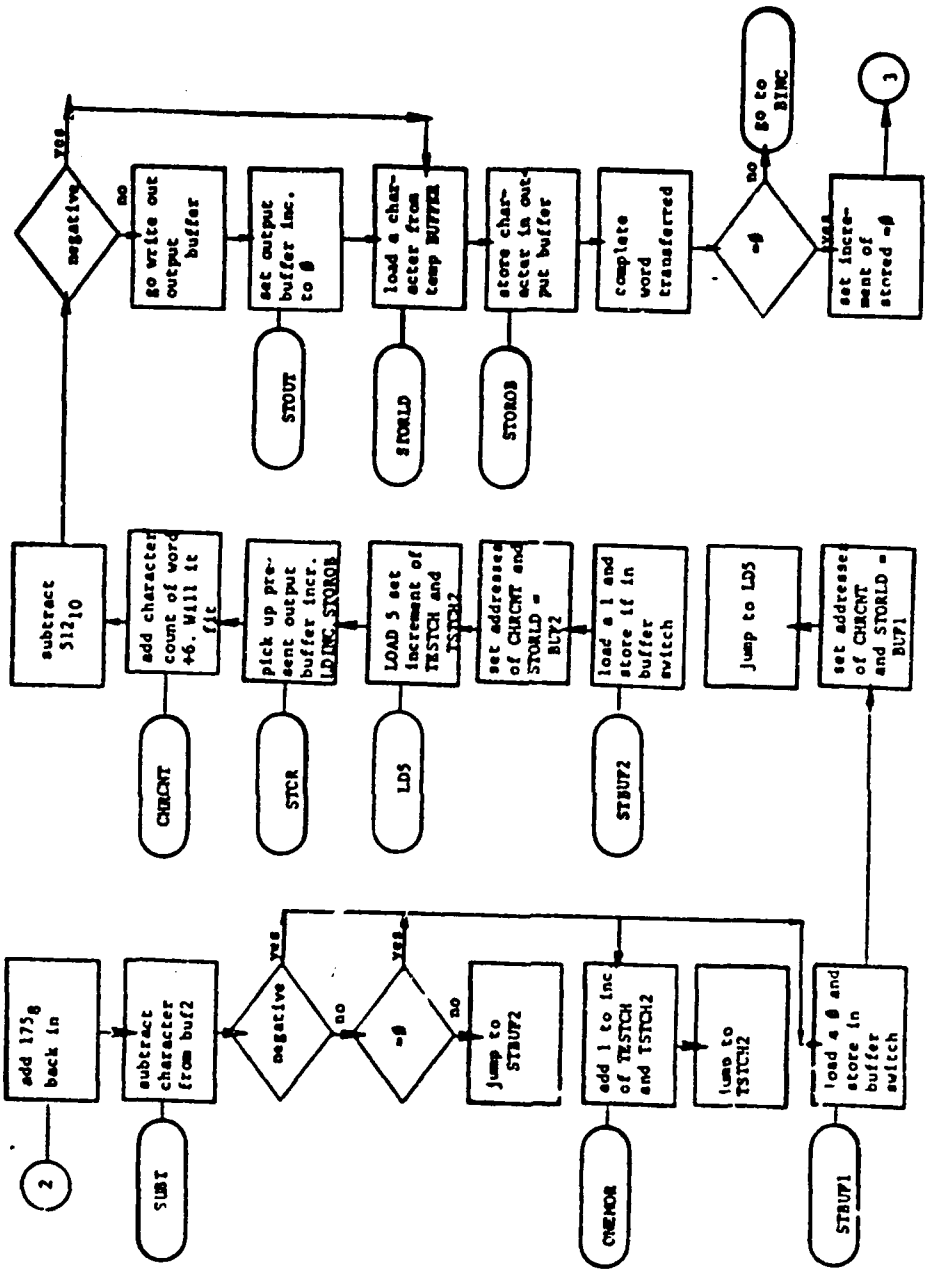

add 175₈ back in
2
subtract character from buf2
SUBT
negative
yes
no
-0
yes
no
jump to STBUF2
add 1 to inc of TESTCH and TSTCH2
CHKCHR
jump to TSTCH2
load a 0 and store in buffer switch
STBUF1
set address of CHKCNT and STORLD - BUF1
jump to LD5
load a 1 and store in buffer switch
STBUF2
set address of CHKCNT and STORLD - BUF2
LOAD 5 set increment of TESTCH and TSTCH2
LD5
pick up present output buffer incr. LDINC, STORDB
STCR
add character count of word +6. Will it fit
CHKCNT
subtract 512₁₀
negative
yes
no
go write out output buffer
set output buffer inc. to 0
STOUT
load a character from temp BUFFER
STORLD
store character in output buffer
STORDB
complete word transferred
-0
yes
no
set increment of stored -0
go to BINC
3

Fig. 4 - 3 (cont'd)

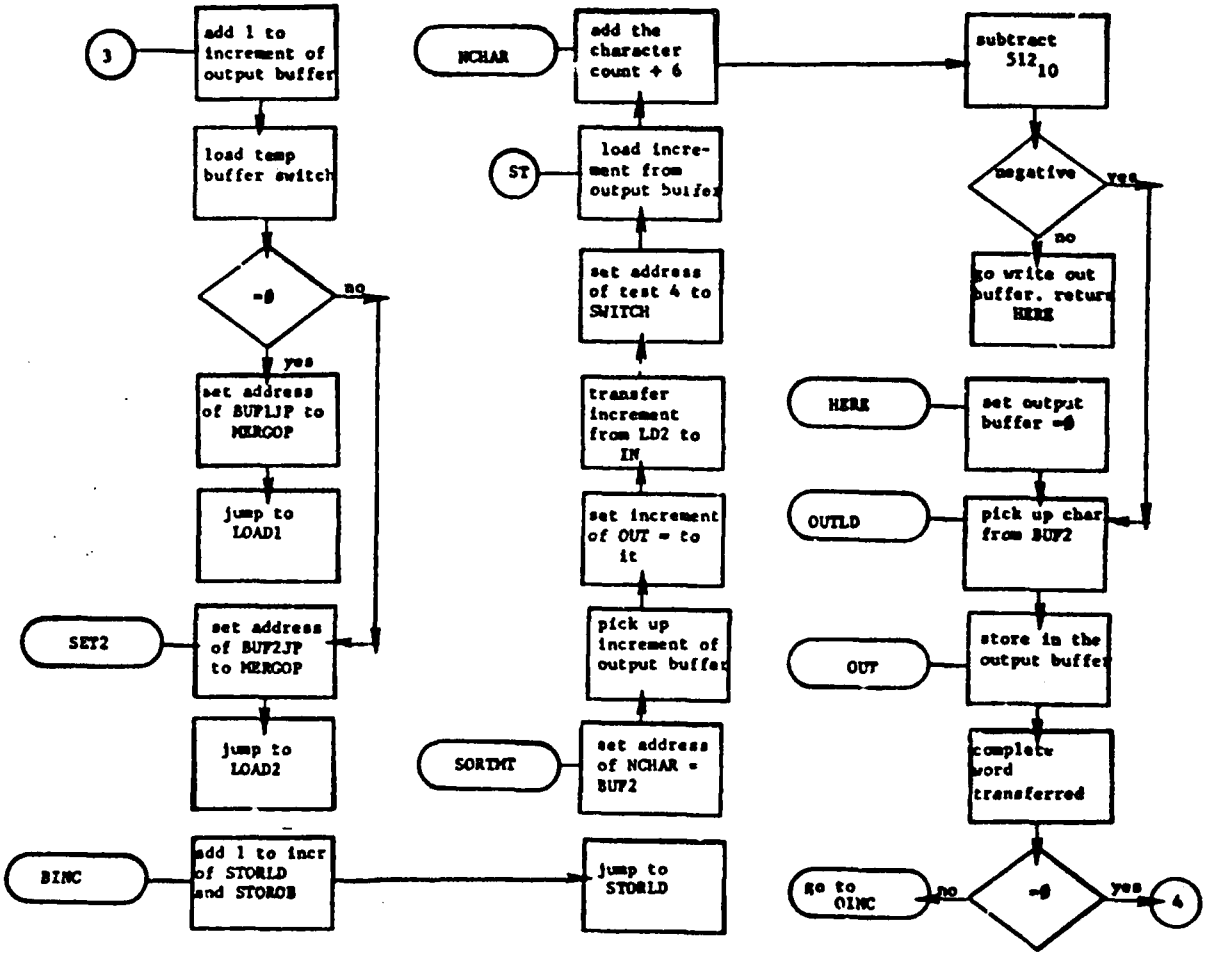

3
add 1 to increment of output buffer
load temp buffer switch
=0
no
yes
set address of BUF1JP to MERGOP
jump to LOAD1
SET2
set address of BUF2JP to MERGOP
jump to LOAD2
BINC
add 1 to incr of STORLD and STOROB
jump to STORLD
SORTMT
set address of NCHAR = BUF2
pick up increment of output buffer
set increment of OUT = to it
transfer increment from LD2 to IN
set address of test 4 to SWITCH
ST
load increment from output buffer
NCHAR
add the character count + 6
subtract 512 10
negative
yes
no
go write out buffer, return HERE
HERE
set output buffer =0
OUTLD
pick up char from BUF2
OUT
store in the output buffer
complete word transferred
go to OINC
no
=0
yes
4

Fig. 4 - 3 (cont'd)

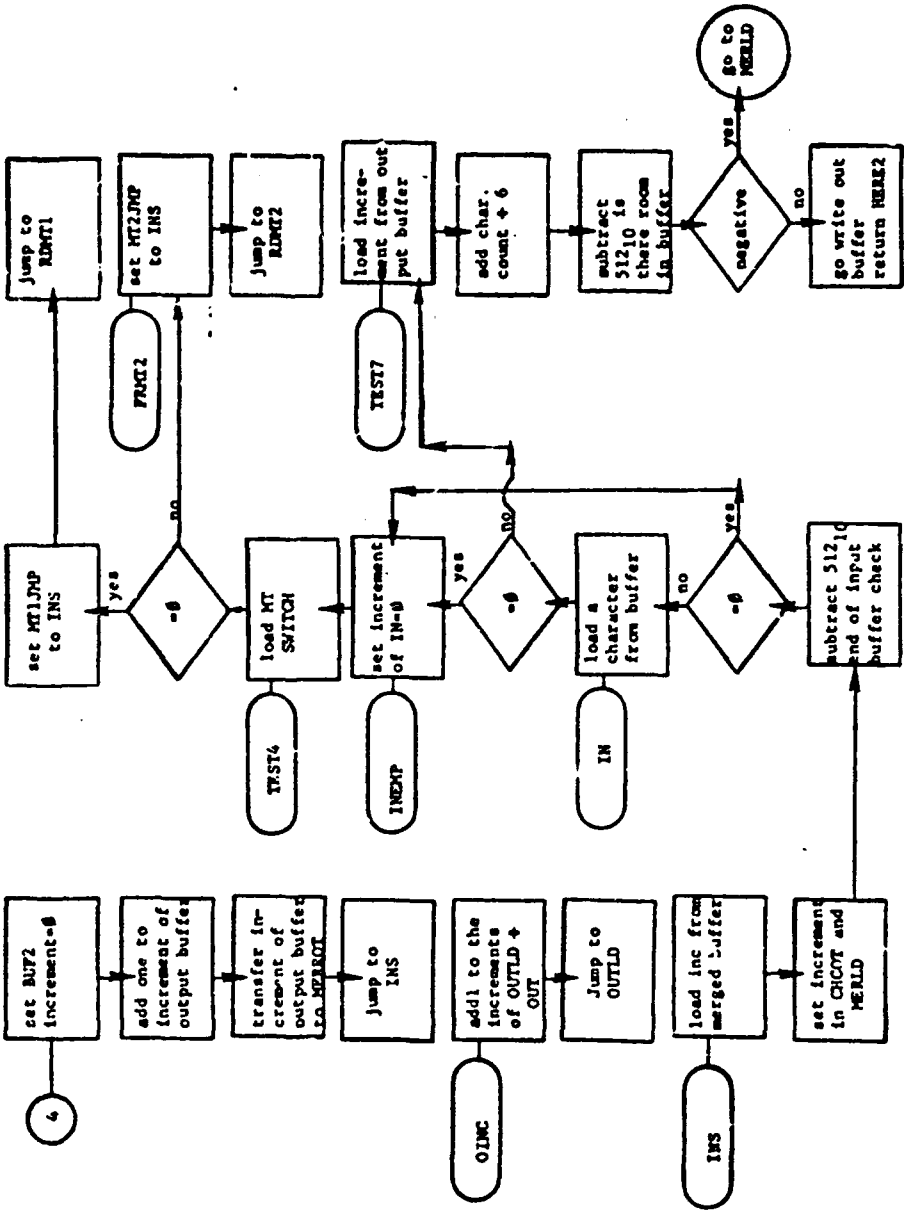

jump to RDMT1
set MT2JMP to INS
jump to RDMT2
load increment from output buffer
add char. count + 6
subtract 512₁₀ is there room in buffer
negative
yes
no
go to MERLD
go write out buffer return HERE2
FMT2
TEST7
set MT1JMP to INS
-0
yes
no
load MT SWITCH
set increment of IN=0
-0
yes
no
load a character from buffer
-0
no
yes
subtract 512,10 end of input buffer check
TEST4
INCMP
IN
set BUF2 increment=0
add one to increment of output buffer
transfer increment of output buffer to MERGOT
jump to INS
add1 to the increments of OUTLD + OUT
Jump to OUTLD
load inc from merged buffer
set increment in CHCNT and MERLD
4
OINC
INS

Fig. 4 - 3 (cont'd)

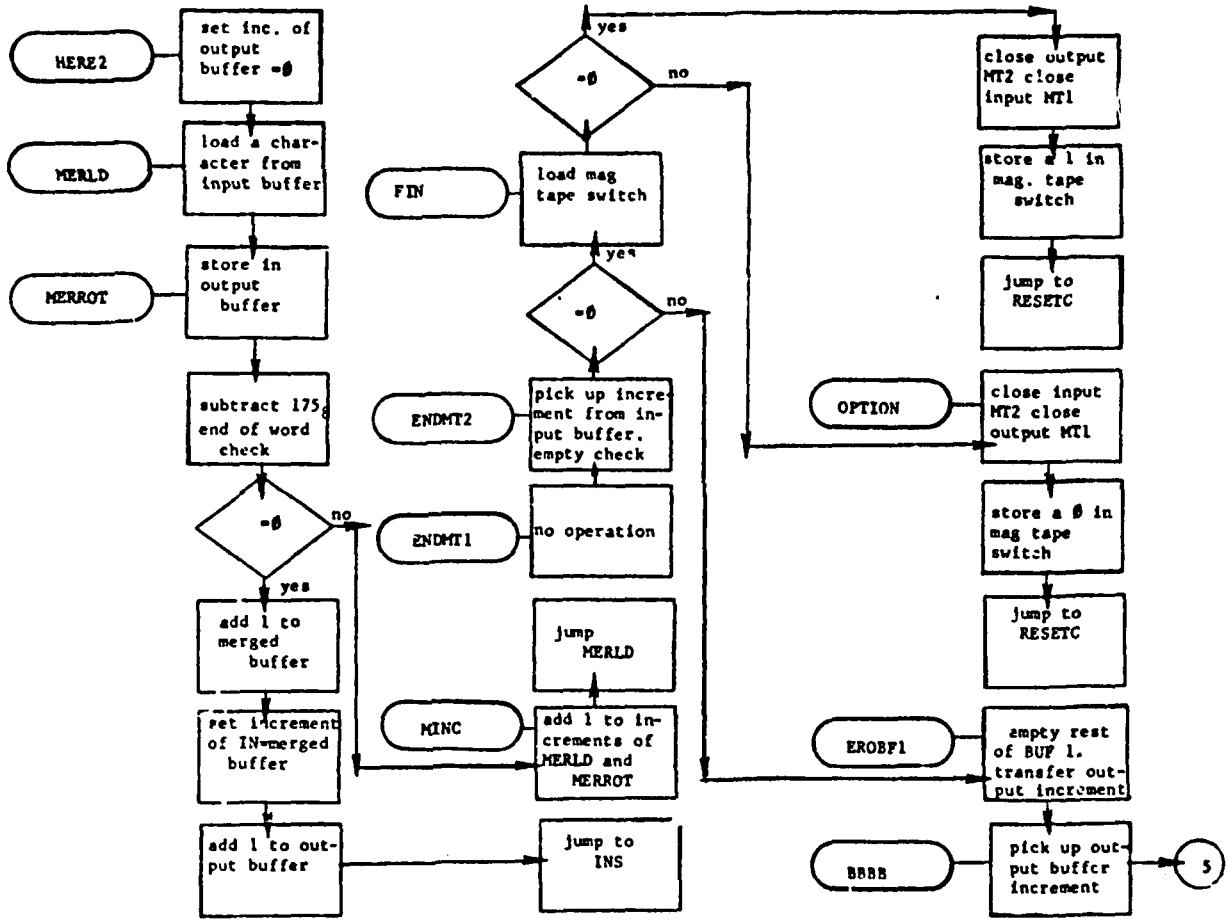

HERE2
set inc. of output buffer =ø
MERLD
load a character from input buffer
MERROT
store in output buffer
subtract 175 end of word check
=ø
no
yes
add 1 to merged buffer
set increment of IN-merged buffer
add 1 to output buffer
jump to INS
MINC
add 1 to increments of MERLD and MERROT
jump MERLD
ENDMT1
no operation
ENDMT2
pick up increment from input buffer. empty check
=ø
yes
no
FIN
load mag tape switch
=ø
yes
no
close output MT2 close input MT1
store a 1 in mag. tape switch
jump to RESETC
OPTION
close input MT2 close output MT1
store a ø in mag tape switch
jump to RESETC
EROBF1
empty rest of BUF 1. transfer output increment
BBBB
pick up output buffer increment
5

Fig. 4 - 3 (cont'd)

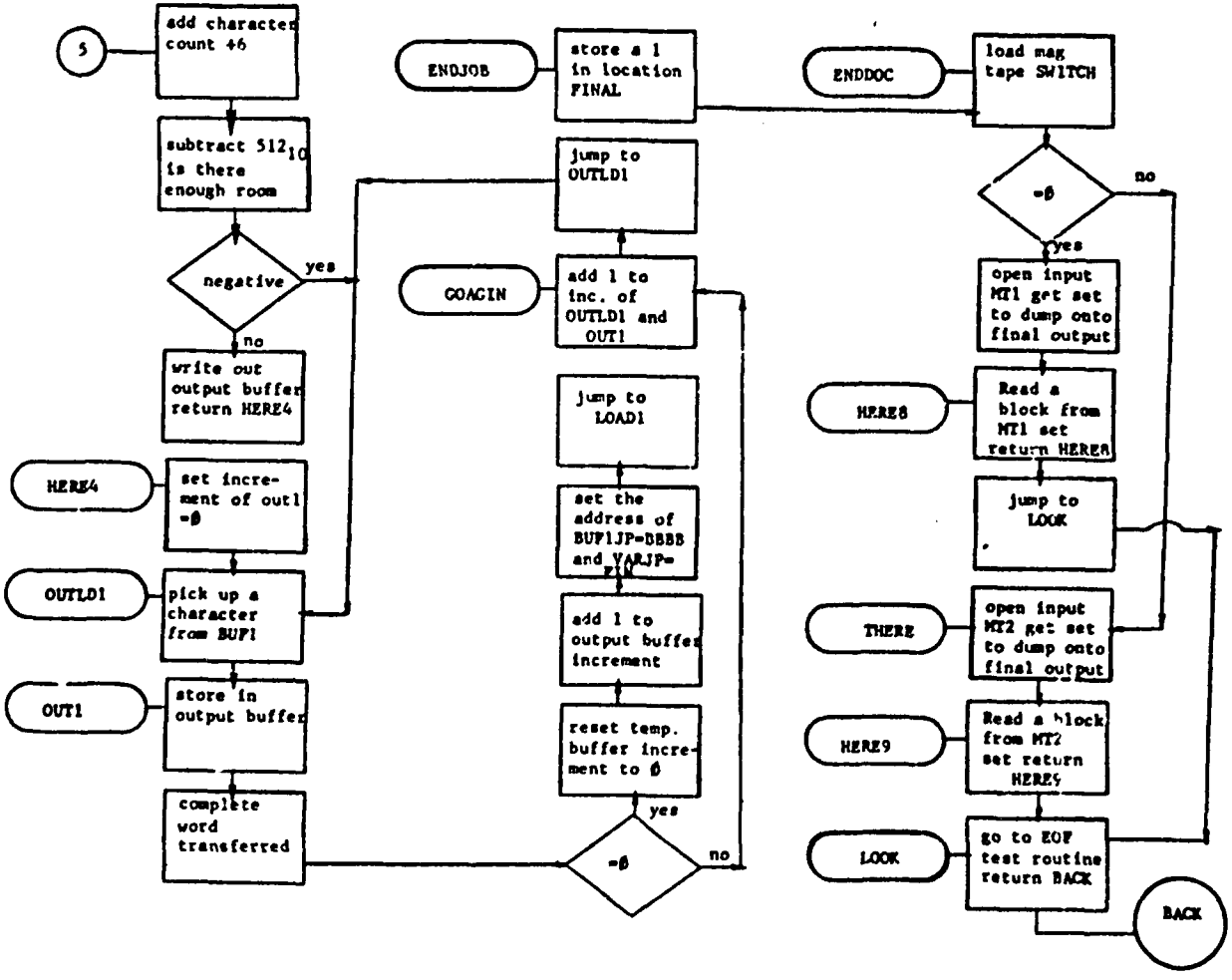

5
add character count 46
subtract 512₁₀ is there enough room
negative
yes
no
write out output buffer return HERE4
HERE4
set increment of out1 =ø
OUTLD1
pick up a character from BUF1
OUT1
store in output buffer
complete word transferred
ENDJOB
store a 1 in location FINAL
jump to OUTLD1
GOAGIN
add 1 to inc. of OUTLD1 and OUT1
jump to LOAD1
set the address of BUF1JP=BBBB and VARJP=
add 1 to output buffer increment
reset temp. buffer increment to ø
=ø
yes
no
ENDDOC
load mag tape SWITCH
=ø
no
yes
open input MT1 get set to dump onto final output
HERE8
Read a block from MT1 set return HERE8
jump to LOOK
THERE
open input MT2 get set to dump onto final output
HERE9
Read a block from MT2 set return HERE9
LOOK
go to EOF test routine return BACK
BACK

Fig. 4 - 3 (cont'd)

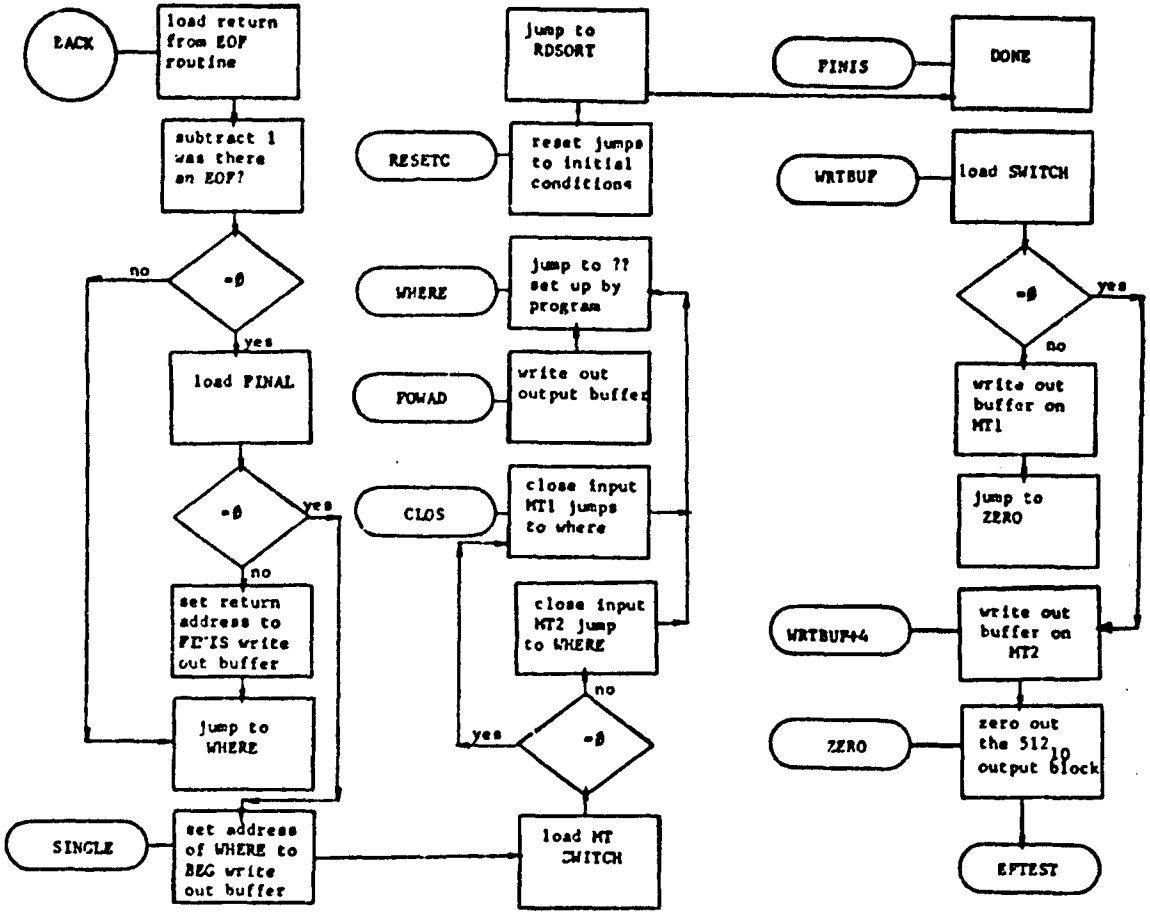

BACK
load return from EOF routine
subtract 1 was there an EOF?
=Ø
no
yes
load FINAL
=Ø
yes
no
set return address to FINIS write out buffer
jump to WHERE
SINGLE
set address of WHERE to BEG write out buffer
jump to RDSORT
RESETC
reset jumps to initial conditions
WHERE
jump to ?? set up by program
POWAD
write out output buffer
CLOS
close input MT1 jumps to where
close input MT2 jump to WHERE
no
yes
=Ø
load MT SWITCH
FINIS
DONE
WRTBUF
load SWITCH
=Ø
yes
no
write out buffer on MT1
jump to ZERO
WRTBUF+4
write out buffer on MT2
ZERO
zero out the 512 output block
EFTEST

Fig. 4 - 3 (cont'd)

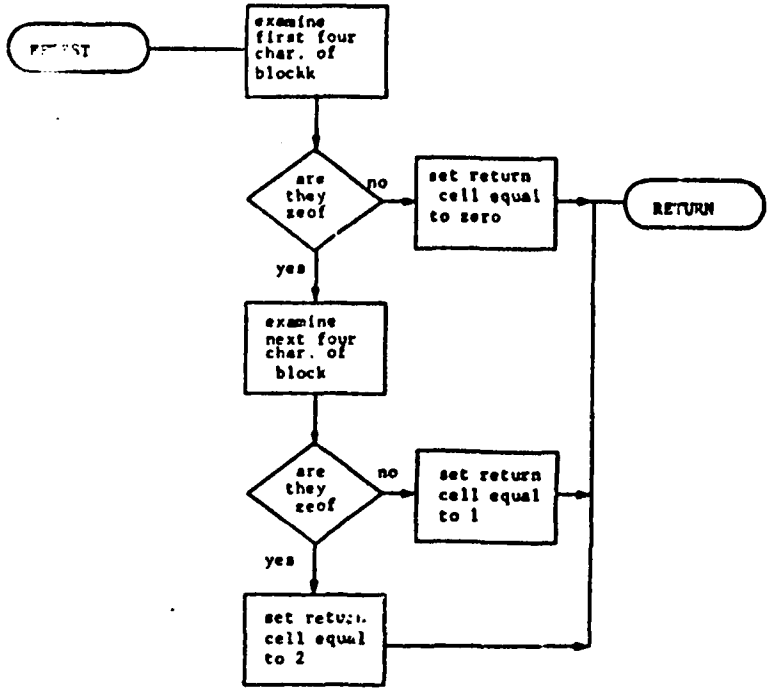

examine
first four
char. of
blockk
are
they
zeof
no
set return
cell equal
to zero
RETURN
yes
examine
next four
char. of
block
are
they
zeof
no
set return
cell equal
to 1
yes
cell equal
to 2

Fig. 4 - 3 (cont'd)

4.4 SHORTWORD (SHRTWD) ROUTINE

4.4.1 Purpose

The purpose of the SHORTWORD routine (see Fig. 4-4) is to correct, in a manner which simulates the shift register comparator logic, all alphabetical words of two or three characters which contain a best guess, a confusion code, or both.

4.4.2 Input

The input tape is the output tape of the SORT routine.

4.4.3 Description

After a block of input has been read, a check is made for an end-of-file block. If either a single (zeof) or a double (zeofzeof) end-of-file block is read, it is written on the output tape. If it is a single end-of-file block, the next block of input is read and processing continues; otherwise, the routing terminates.

If it is not an end-of-file block, it is a data block and the first word of each item, the character count, is examined to see whether it is two or three. If the character count is not two or three, the item is stored, as is, in the output buffer and the routine continues to process data words until an item with a character count of zero is encountered. At this point, the output buffer is written on the output tape and the next block is read.

If the character count is two or three, the second word of the item, the type code, is examined. If it indicates that the word contains no confusion or best guess codes, the item is stored, as is, in the output buffer. However, if the type code does contain an indication of one or both codes, then it is necessary to compare the data word portion of the item with the shortword dictionary, which is stored in memory, in order to see if the word is correct or if it is necessary to correct it. If an exact match is obtained between the lookup word and a dictionary entry, the word is stored, as is, in the output buffer.

If an exact match is not obtained during the first examination, as is the case when the word contains a confusion character, then one of the characters is disregarded and a match of the remaining characters is sought. If obtained, the corrected word is stored with its threshold set to one, indicating that the word has been corrected but it was necessary to disregard one character in order to correct it. This threshold value is placed in the five leftmost bits of the fifth word of the item. The left most bit indicates whether the word has been corrected (0) or not (1). The other four bits contain the threshold level used in correcting the word.

If a data word is not corrected and it is a two-letter word, the threshold value has reached a maximum for this length word and it is considered not correctable. Hence, the fifth word is flagged to indicate an uncorrected word with a threshold value of one. However, if it is a three-letter data word,

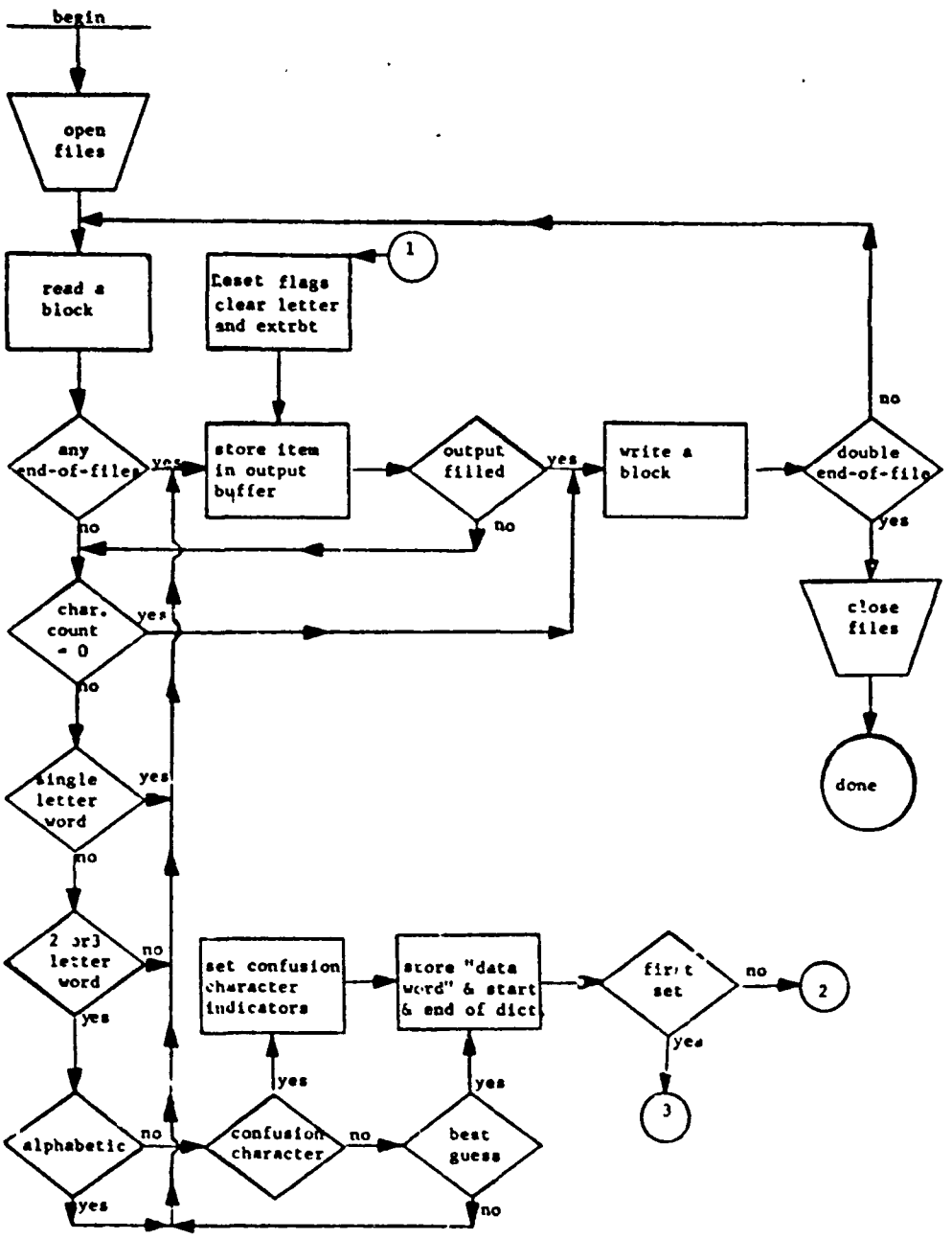


Fig. 4 - 4 Flow diagram for Shortword routine

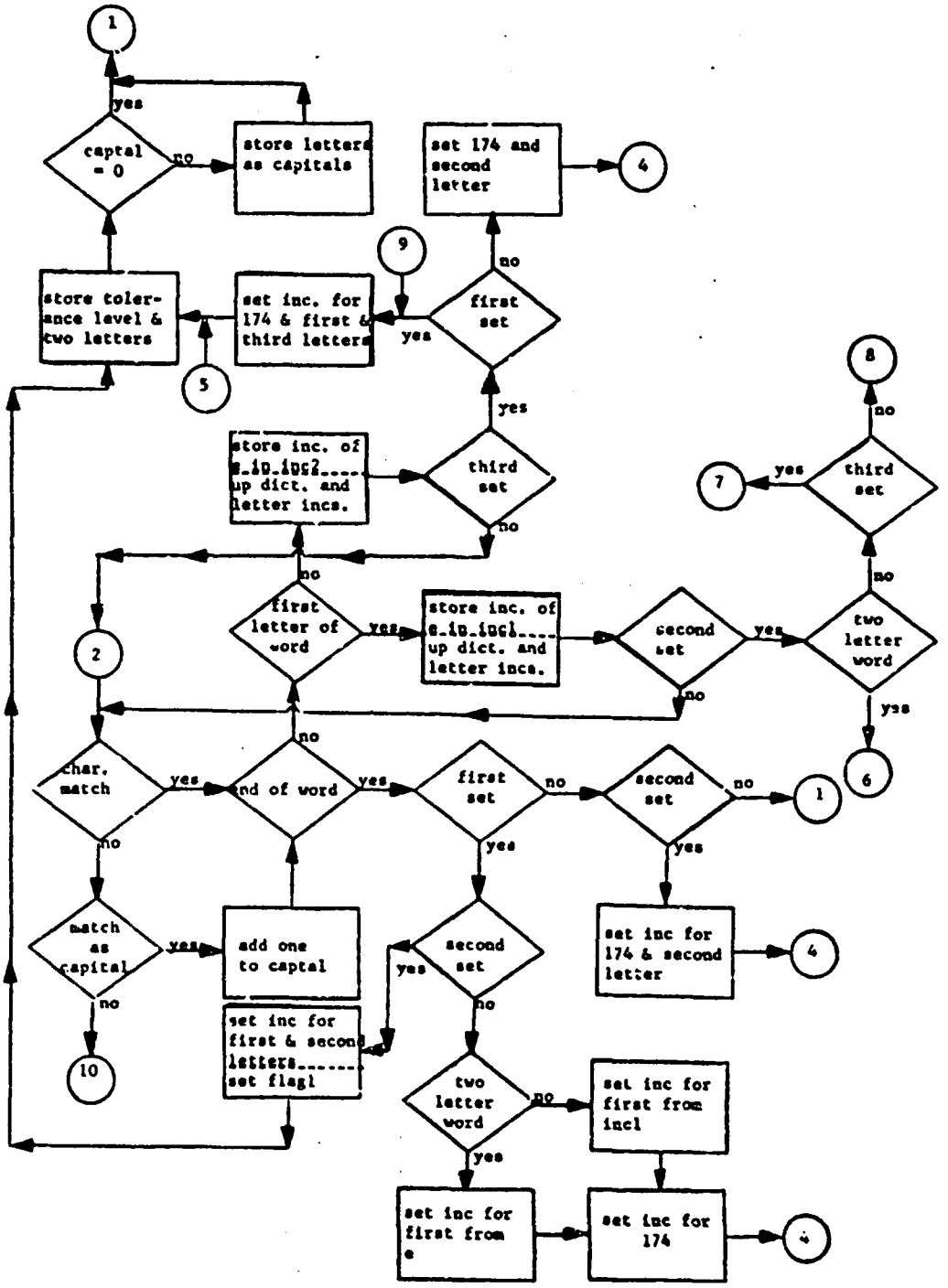

1
yes
captal = 0
no
store letters as capitals
set 174 and second letter
4
9
no
first set
yes
store tolerance level & two letters
set inc. for 174 & first & third letters
5
yes
store inc. of g_in_inc2 up dict. and letter incs.
third set
no
8
no
7
yes
third set
no
2
no
first letter of word
yes
store inc. of g_in_incl up dict. and letter incs.
second set
yes
two letter word
no
yes
6
no
char. match
yes
end of word
yes
first set
no
second set
no
1
no
yes
yes
match as capital
yes
add one to captal
second set
set inc for 174 & second letter
4
no
yes
set inc for first & second letters set flag1
10
two letter word
no
set inc for first from incl
yes
set inc for first from e
set inc for 174
4

Fig. 4 - 4 (cont'd)

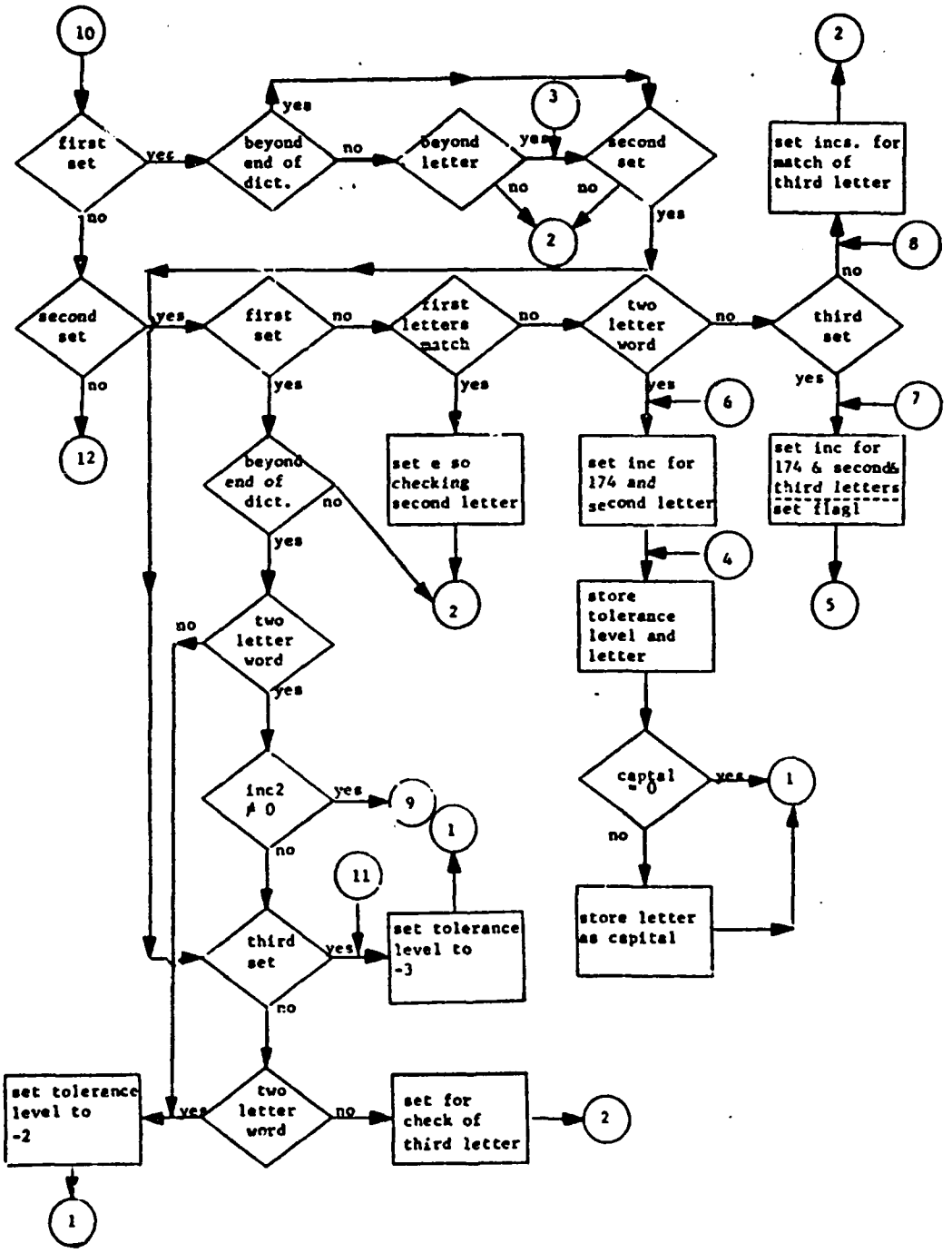

10
first set
yes
beyond end of dict.
yes
no
beyond letter
yes
3
second set
no
no
2
yes
2
set incs. for match of third letter
8
no
second set
yes
first set
no
first letters match
no
two letter word
no
third set
yes
7
no
12
yes
yes
yes
yes
6
beyond end of dict.
no
set e so checking second letter
set inc for 174 and second letter
set inc for 174 & seconds third letters
set flag1
yes
4
2
store tolerance level and letter
5
no
two letter word
yes
capital ≠ 0
yes
1
inc2 ≠ 0
yes
9
1
no
no
11
store letter as capital
third set
yes
set tolerance level to -3
no
set tolerance level to -2
yes
two letter word
no
set for check of third letter
2
1

Fig. 4 - 4 (cont'd)

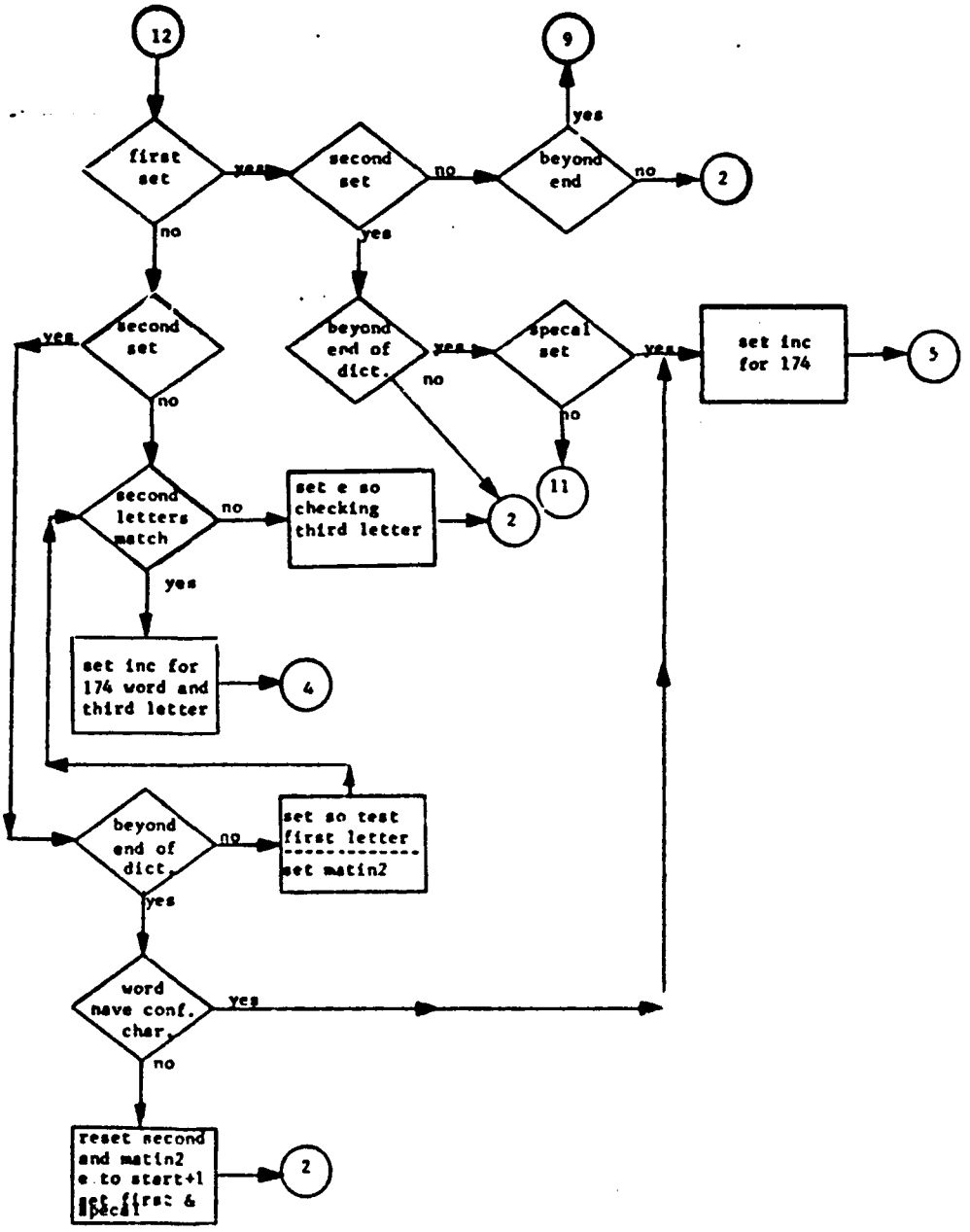

12
first set
yes
second set
no
9
yes
beyond end
no
2
no
yes
second set
yes
no
beyond end of dict.
yes
no
special set
yes
no
set inc for 174
5
second letters match
no
set e so checking third letter
2
11
yes
set inc for 174 word and third letter
4
beyond end of dict.
no
set so test first letter
set matin2
yes
word have conf. char.
yes
no
reset second and matin2
e to start+1
set first &
special
2

Fig. 4 - 4 (cont'd)

another character is disregarded and a match is again sought. If a successful match is made, the word is stored, as is, with the fifth word indicating an uncorrected word with a threshold level of two.

After every item is stored, a check is made to see if the output buffer is completely full. If it is full, the output buffer is written and the next input block is read.

4.4.4 Output

The output tape has exactly the same format and order as the input tape. The only differences are that flags have been added to indicate threshold levels and confusion and best guess codes have been replaced in those words that could be corrected.

4.5 LOAD DISK (LODDSK) ROUTINE

4.5.1 Purpose

The purpose of the LOAD DISK routine (see Fig. 4-5) is to load a dictionary onto the BD-500 disk, and, while performing the loading operation, to gather specific information about where the dictionary is stored on the disk. The dictionary storage information is saved in a reserved area which is accessible by other routines. The primary user of the information is the routine which drives the shift register comparator. After determining various characteristics about a "word" to be matched or corrected, the SRC routine selects the appropriate dictionary segment from the information in the reserved area. The LOAD DISK routine also adds various flags necessary for proper functioning of the shift register comparator status responses.

4.5.2 Input

The input is a paper tape containing the appropriate dictionary. The tape should be punched in bioctal codes. The dictionary words should be grouped by character length N, in ascending order, where N≤4 and should be alphanumerically sorted within each character length. This sequence is required for the shift register comparator to function properly.

4.5.3 Description

The reserved area (DICTAB) is initialized at BANK 1 and TRACK 1, which is the address of the first load. The dictionary is read in a block (512_{10} characters) at a time. This block is then transferred to an output buffer (814_{10} characters) word by word. (The size of the output buffer is prescribed in ECP 2 Technical Note 56, Supplement 1.) This enables the routine to load the disk in the two-track mode. As each word is transferred to the output buffer, two checks are made. The first is to allow only complete words to be transferred. The total character count, calculated by summing the individual character counts of the transferred words, should not exceed 813_{10}. This leaves the required one location for inserting the octal code 173, which is a flag required by the shift register comparator. The second check is to note any change in character length. Upon finding a change, no more words are transferred and the output buffer is considered full. Then the octal code 173 is

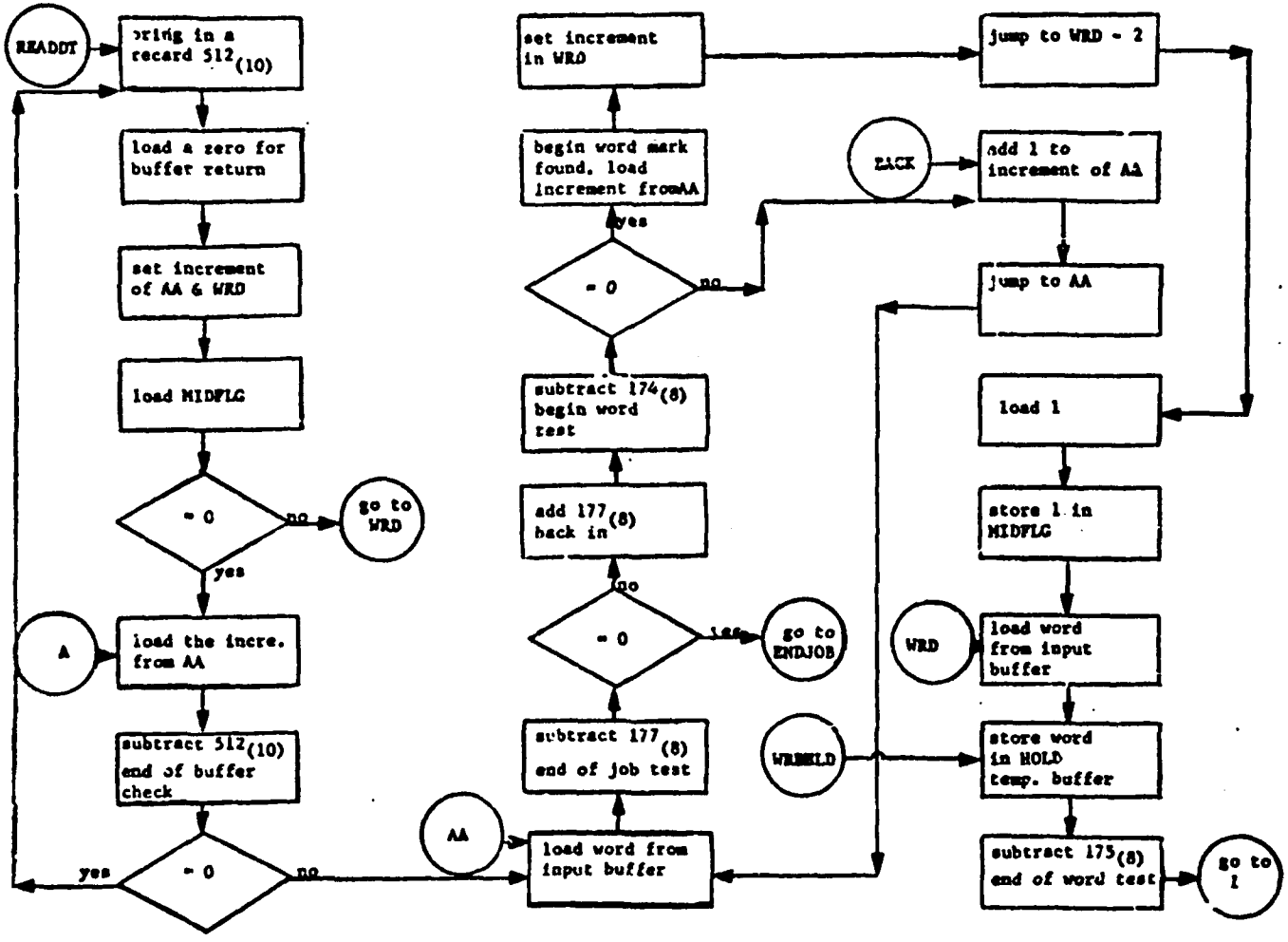


Fig. 4 - 5 Flow diagram for Load Disk routine

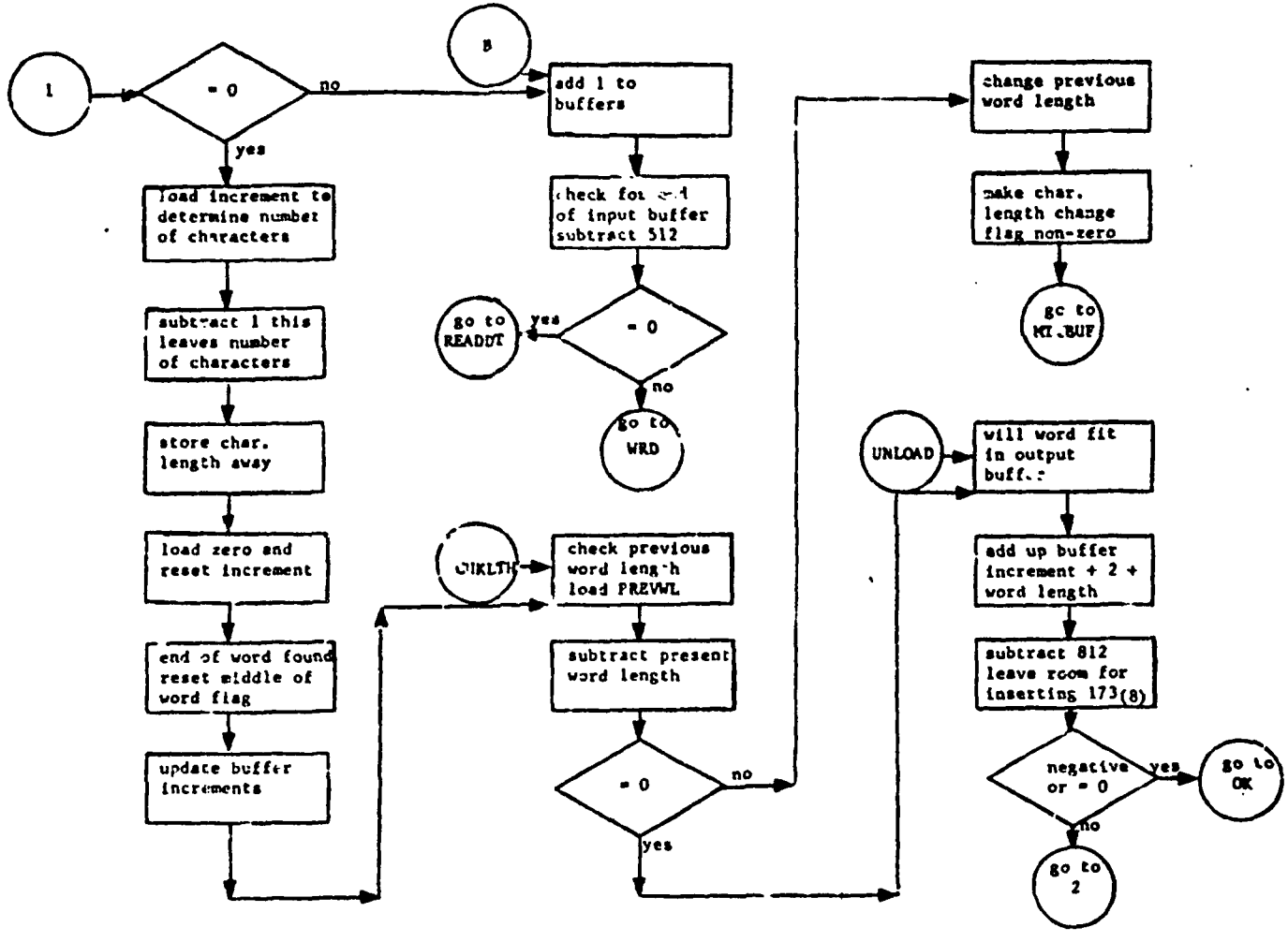

1
= 0
no
yes
load increment to determine number of characters
subtract 1 this leaves number of characters
store char. length away
load zero and reset increment
end of word found reset middle of word flag
update buffer increments
B
add 1 to buffers
check for end of input buffer subtract 512
= 0
yes
go to READDT
no
go to WRD
CHKLTH
check previous word length load PREVWL
subtract present word length
= 0
no
yes
change previous word length
make char. length change flag non-zero
go to MT.BUF
UNLOAD
will word fit in output buff.?
add up buffer increment + 2 + word length
subtract 812 leave room for inserting 173(8)
negative or = 0
yes
go to OK
no
go to 2

Fig. 4 - 5 (cont'd)

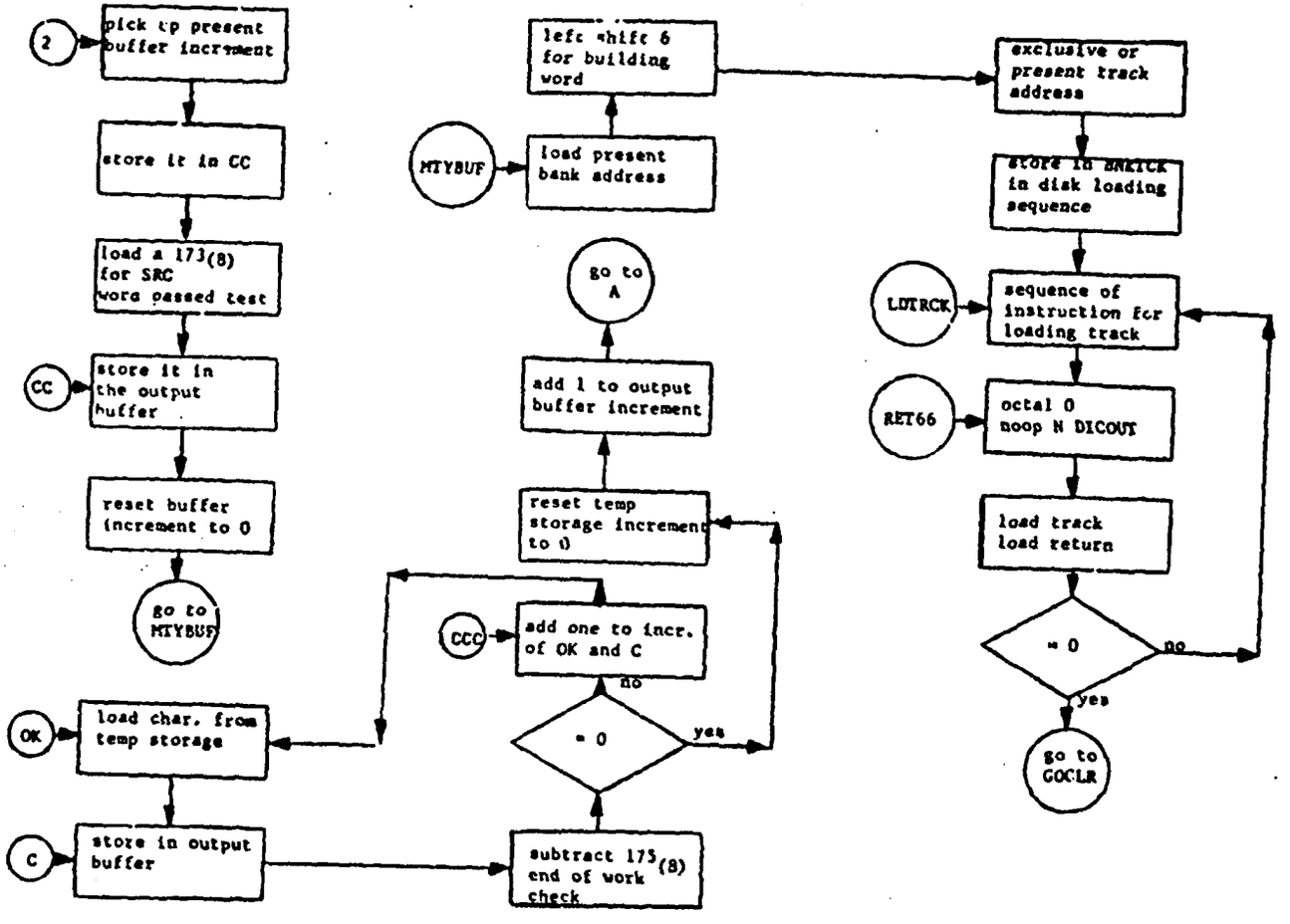

2
pick up present buffer increment
store it in CC
load a 173(8) for SRC word passed test
CC
store it in the output buffer
reset buffer increment to 0
go to MTYBUF
OK
load char. from temp storage
C
store in output buffer
subtract 173(8) end of work check
= 0
yes
no
CCC
add one to incr. of OK and C
reset temp storage increment to 0
add 1 to output buffer increment
go to A
MTYBUF
load present bank address
left shift 6 for building word
exclusive or present track address
store in BNKTCK in disk loading sequence
LDTRCK
sequence of instruction for loading track
RET66
octal 0 noop N DICOUT
load track load return
= 0
no
yes
go to GOCLR

Fig. 4 - 5 (cont'd)

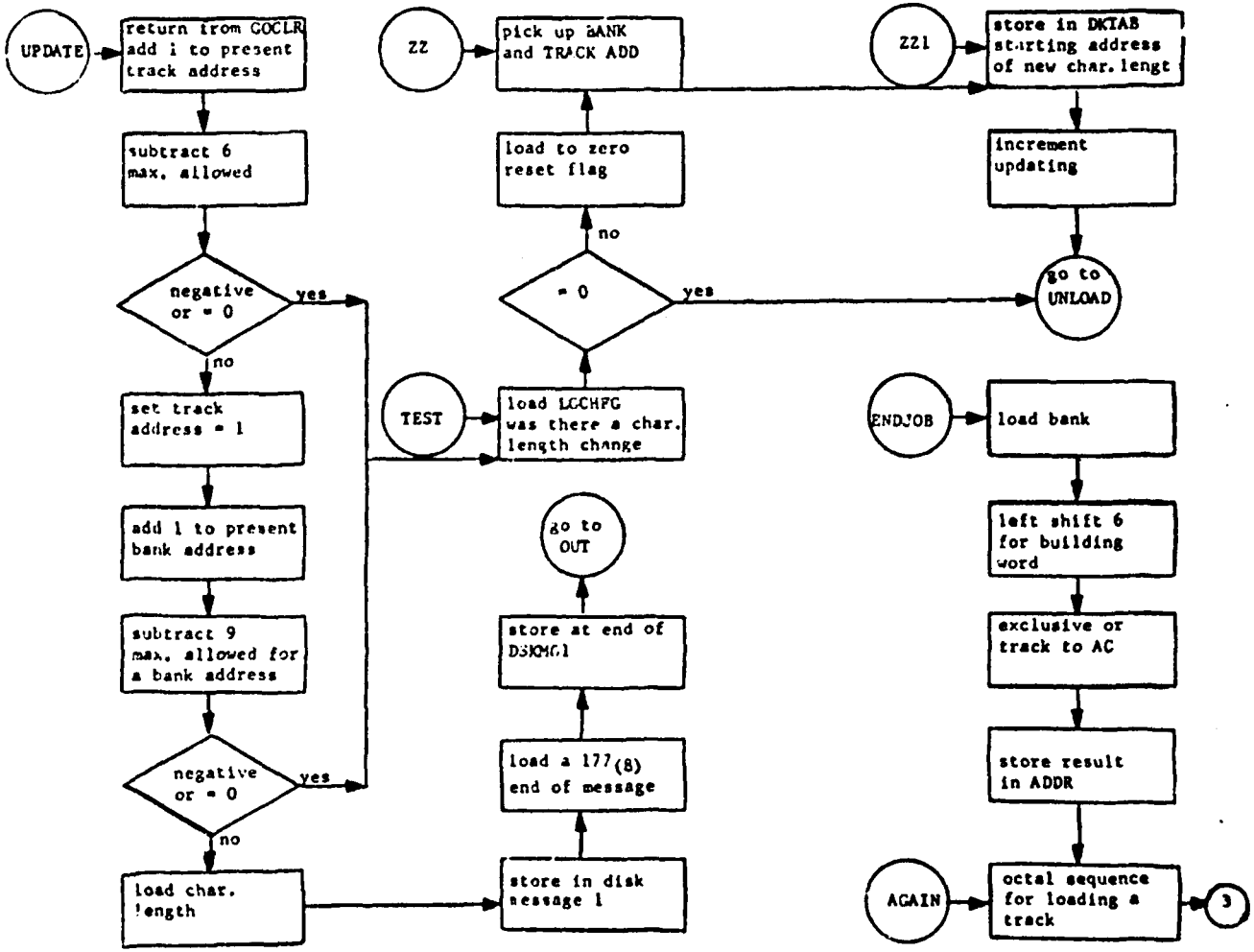

UPDATE
return from GOCLR
add 1 to present
track address
subtract 6
max. allowed
negative
or = 0
yes
no
set track
address = 1
add 1 to present
bank address
subtract 9
max. allowed for
a bank address
negative
or = 0
yes
no
load char.
length
store in disk
message 1
load a 177(8)
end of message
store at end of
DSKMG1
go to
OUT
TEST
load LGCHFG
was there a char.
length change
load to zero
reset flag
= 0
no
yes
ZZ
pick up BANK
and TRACK ADD
ZZ1
store in DKTAB
starting address
of new char. lengt
increment
updating
go to
UNLOAD
ENDJOB
load bank
left shift 6
for building
word
exclusive or
track to AC
store result
in ADDR
AGAIN
octal sequence
for loading a
track
3

Fig. 4 - 5 (cont'd)

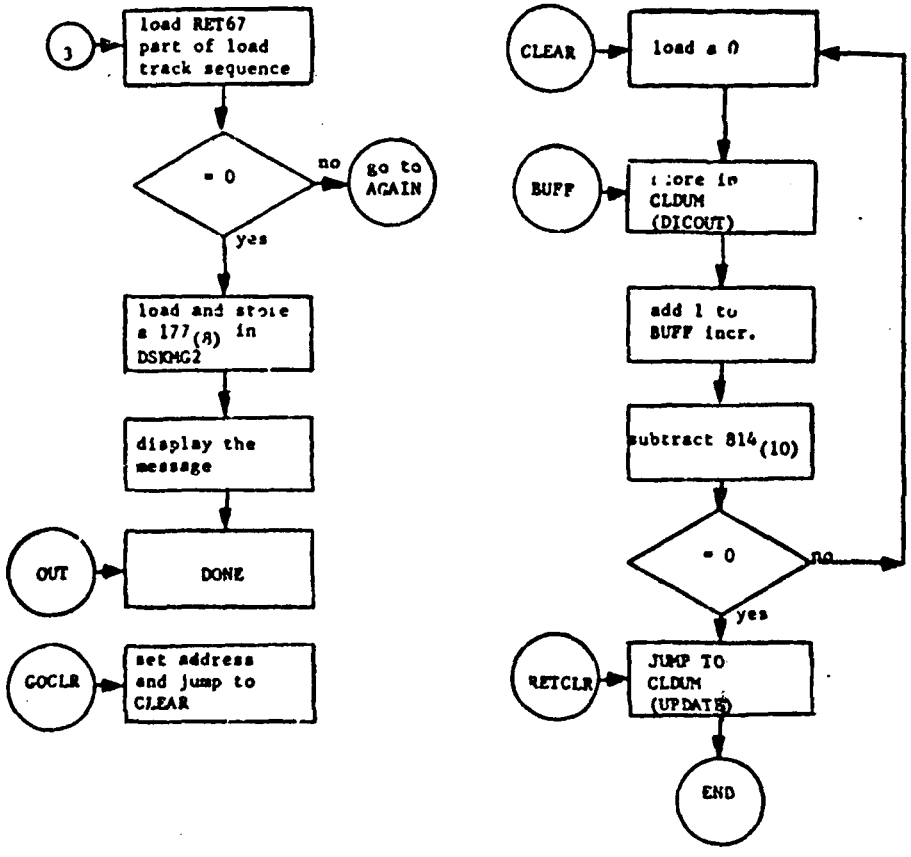

3
load RET67
part of load
track sequence
= 0
no
go to
AGAIN
yes
load and store
a 177(8) in
DSKMG2
display the
message
OUT
DONE
GOCLR
set address
and jump to
CLEAR
CLEAR
load a 0
BUFF
store in
CLDUM
(DICOUT)
add 1 to
BUFF incr.
subtract 814(10)
= 0
no
yes
RETCLR
JUMP TO
CLDUM
(UPDATE)
END

Fig. 4 - 5 (cont'd)

inserted. When either of these conditions is encountered the buffer is loaded onto the disk, the BANK and TRACK address is updated, the output buffer is zeroed out, and processing continues. However, if a character length change is encountered, the latest BANK and TRACK address is entered into the reserved area before continuing. Upon completion of this routine a message is typed out on the Selectric typewriter to the effect that the dictionary has been successfully loaded.

4.5.4 Output

The output of this routine is the dictionary loaded on the BD-500 disk and a table which contains the starting address (BANK and TRACK) of each character length segment of the stored dictionary.

4.6 SHIFT REGISTER COMPARATOR (SRC) ROUTINE

4.6.1 Purpose

The purpose of the SRC routine (see Fig. 4-6) is to examine the data previously sorted in alphanumeric order and flagged according to data word type and to set the shift register comparator conditions for a dictionary lookup. An attempt is then made either to match or to correct the word. The final result is flagged accordingly.

4.6.2 Input

The input is a magnetic tape containing N batches of M documents, where a document is composed of one or more blocks of 512 computer words. Each batch of M documents is sorted in ascending order of character length and alphanumerically within each character length for the data words in the batch.

Each individual item (see Appendix A) has six computer words added to it for descriptive purposes. This program is concerned with the first two of the descriptive words in an item. Word 1 contains the character count of the item to be processed. Word 2 indicates the classification of the item (see Appendix A for an explanation of type codes). A single end-of-file block (zeof in the first four words of 512-word block) denotes the logical break between batches of documents. A double end-of-file block (zeofzeof in the first eight words of a 512-word block) denotes the end of the job.

4.6.3 Description

For brevity the shift register comparator will be referred to as the SRC and its individual registers will be referred to as SR1, SR2, SR3, and SR4. (See SRC hardware manual for details.)

Data is read in one block at a time. One item is then transferred to a buffer area for editing. The first check is the character count; if it is less than 4 or greater than 18, the item is bypassed. This limitation is brought about by the actual SRC hardware configuration and operation. The type classification is then examined. If the item is anything but alphabetic, it is bypassed as being uncorrectable. If both of these tests are passed, then the

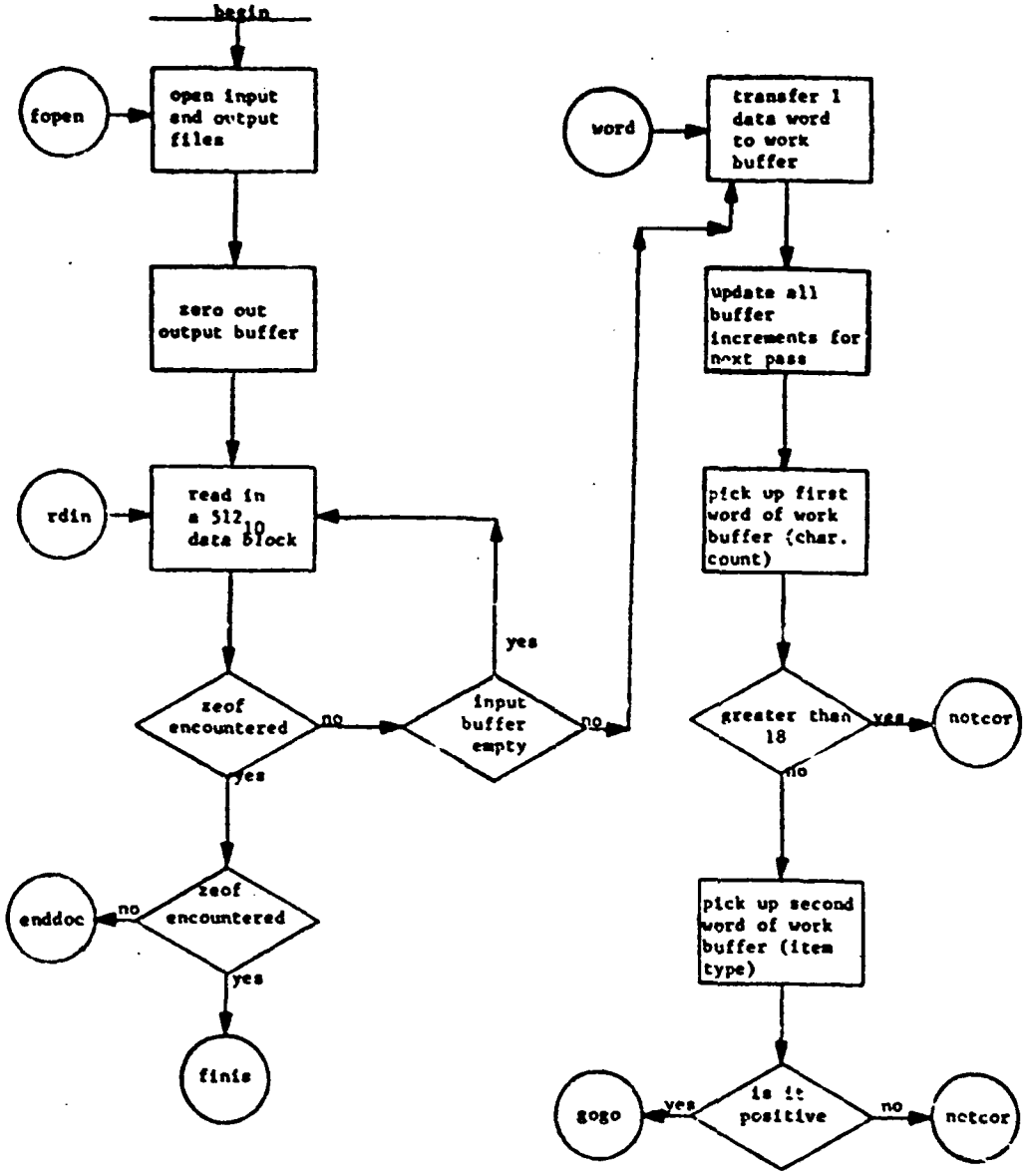


Fig. 4 - 6 Flow diagram for Shift-Register Comparator routine

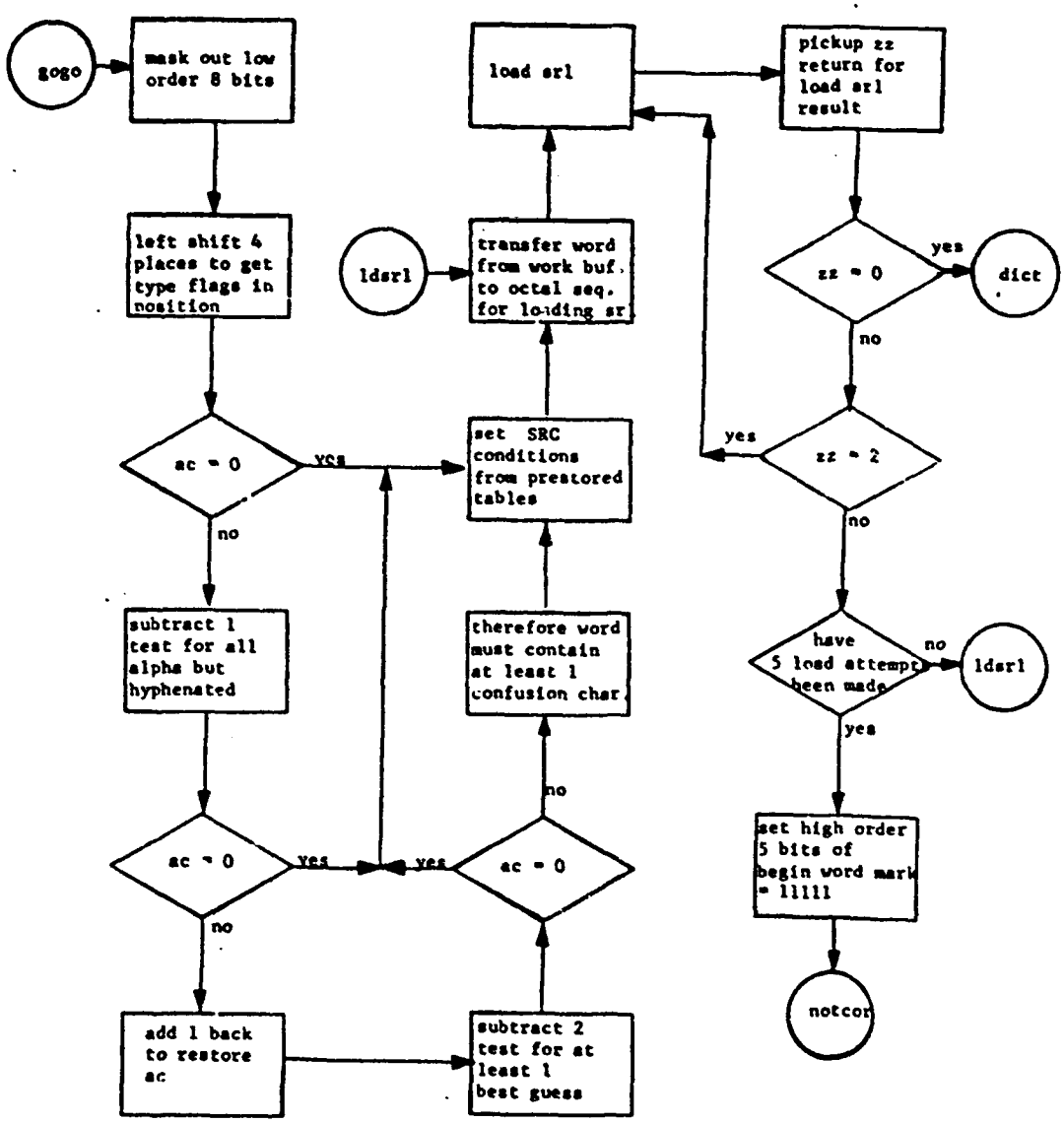

gogo
mask out low order 8 bits
left shift 4 places to get type flags in position
ac = 0
yes
no
subtract 1 test for all alpha but hyphenated
ac = 0
yes
no
add 1 back to restore ac
ldsrl
transfer word from work buf. to octal seq. for loading sr
load srl
set SRC conditions from prestored tables
therefore word must contain at least 1 confusion char
yes
ac = 0
no
subtract 2 test for at least 1 best guess
pickup zz return for load srl result
zz = 0
yes
dict
no
yes
zz = 2
no
have 5 load attempts been made
no
ldsrl
yes
set high order 5 bits of begin word mark = 11111
notcor

Fig. 4 - 6 (cont'd)

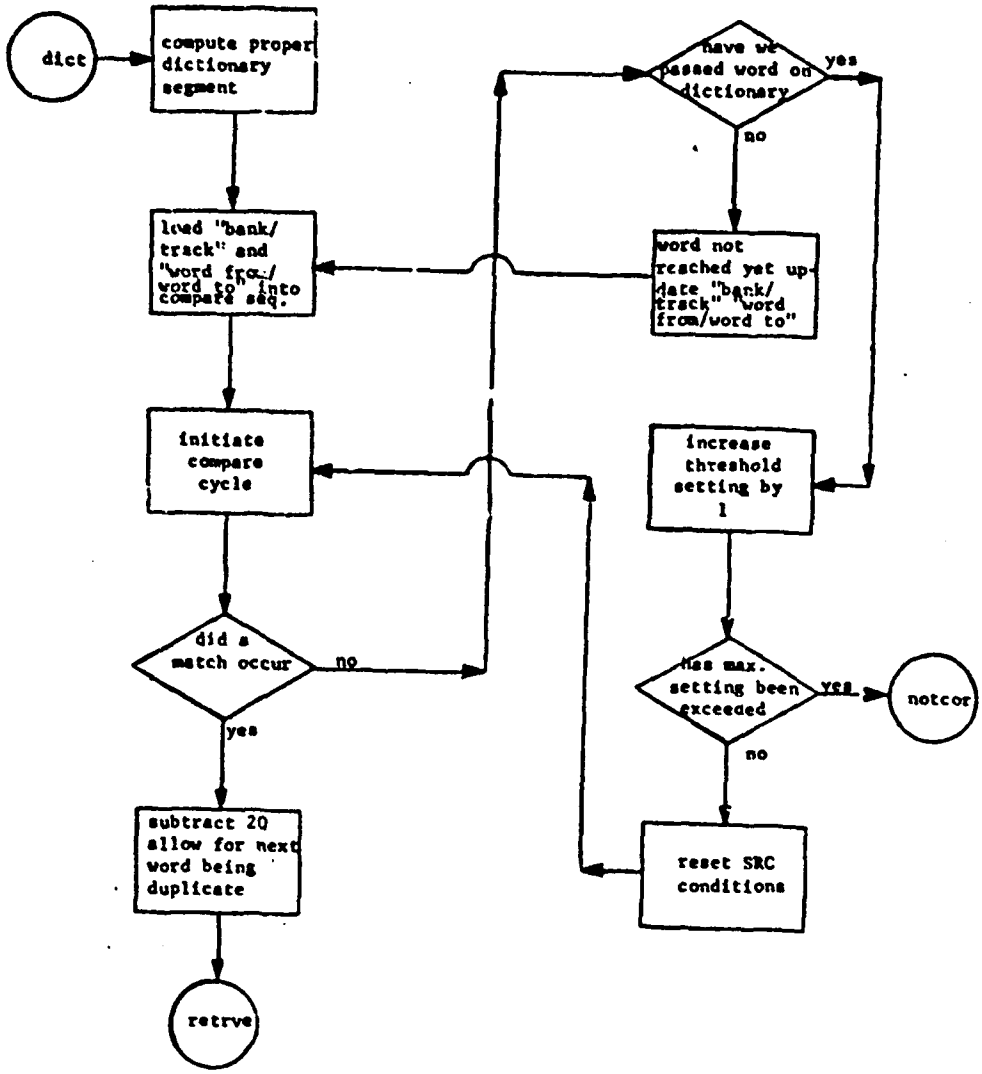

dict
compute proper dictionary segment
load "bank/ track" and "word fro-/ word to" into compare seq.
initiate compare cycle
did a match occur
no
yes
subtract 20 allow for next word being duplicate
retrve
have we passed word on dictionary
yes
no
word not reached yet up-date "bank/ track" "word from/word to"
increase threshold setting by 1
has max. setting been exceeded
yes
notcor
no
reset SRC conditions

Fig. 4 - 6 (cont'd)

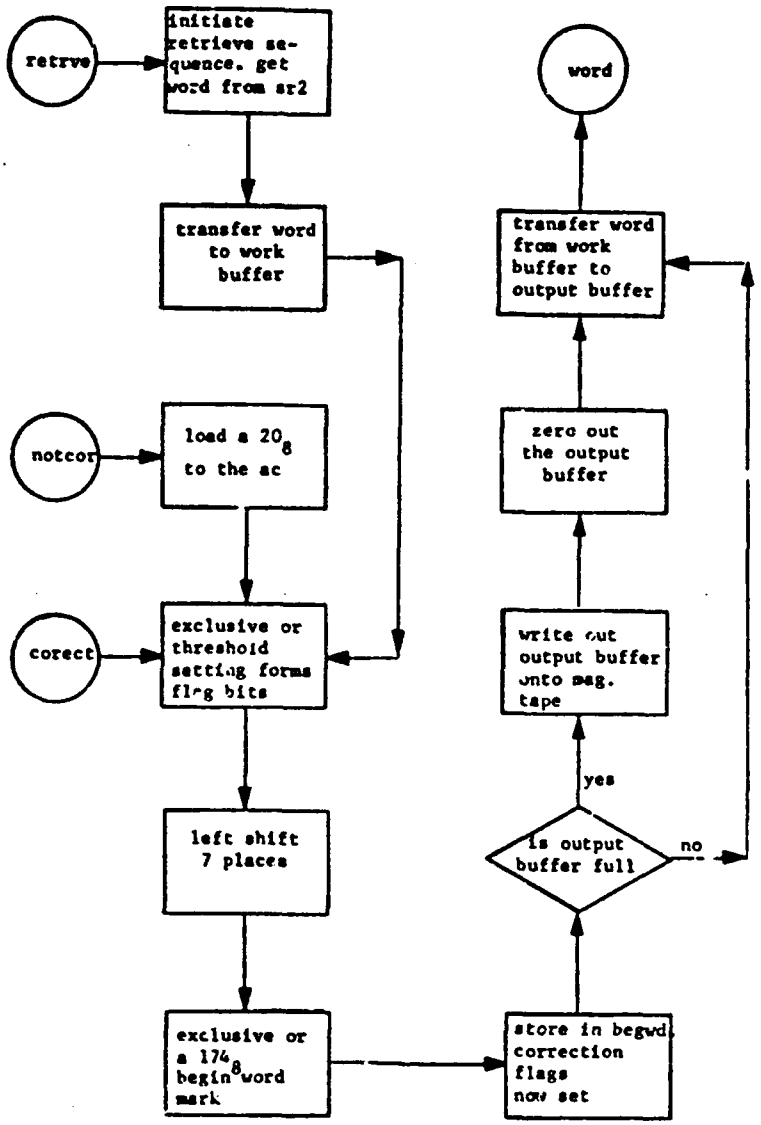

retrve
initiate retrieve sequence. get word from sr2
transfer word to work buffer
notcor
load a 20₈ to the ac
corect
exclusive or threshold setting forms flag bits
left shift 7 places
exclusive or a 174₈ begin word mark
store in begwd. correction flags now set
is output buffer full
yes
no
write out output buffer onto mag. tape
zero out the output buffer
transfer word from work buffer to output buffer
word

Fig. 4 - 6 (cont'd)

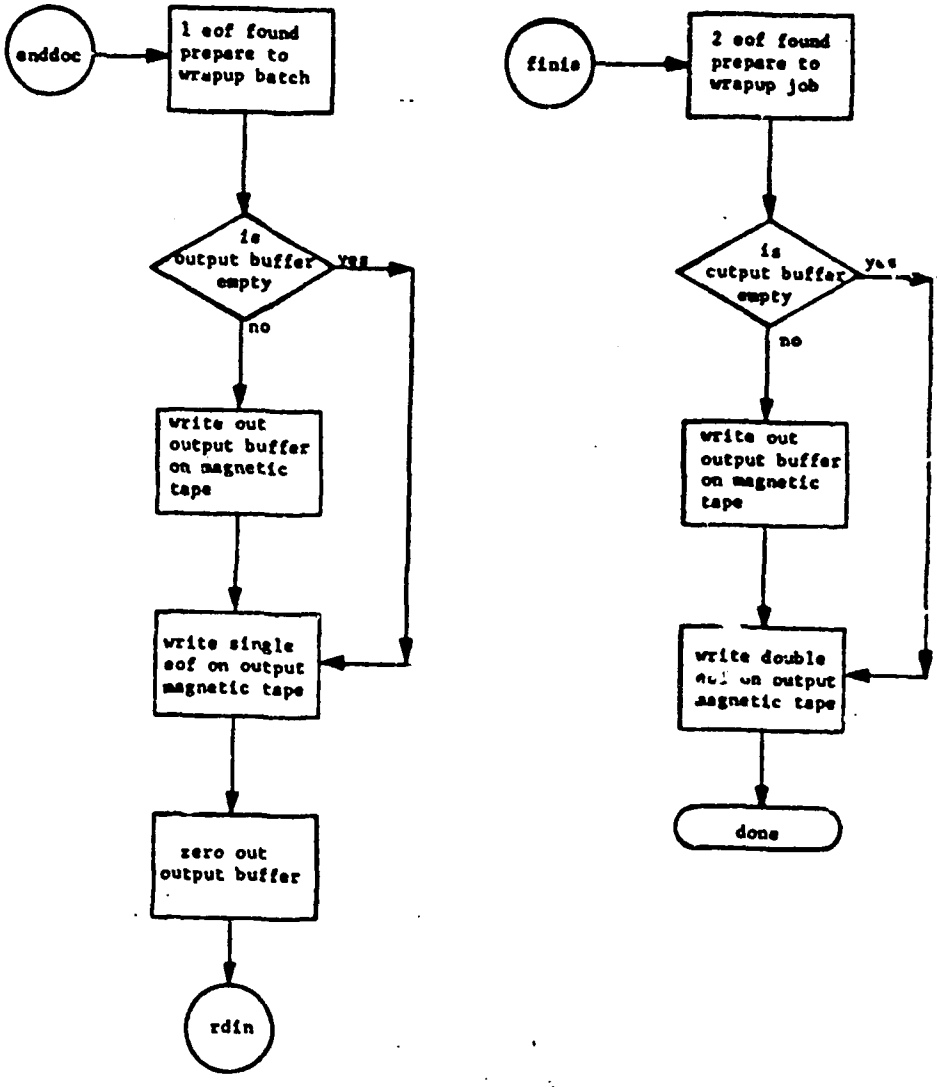

enddoc
1 eof found prepare to wrapup batch
is output buffer empty
yes
no
write out output buffer on magnetic tape
write single eof on output magnetic tape
zero out output buffer
rdin
finis
2 eof found prepare to wrapup job
is output buffer empty
yes
no
write out output buffer on magnetic tape
write double eof on output magnetic tape
done

Fig. 4 - 6 (cont'd)

SRC conditions are set according to type found: alphabetic, alphabetic with best guess, alphabetic with confusion, alphabetic with confusion plus best guess. (See Appendix D for bit configuration of settings which can be set and then lowered to allow for the best possible match.) If a data word has been determined as containing all alphabetic characters, the threshold is set at zero. The data word plus beginning and end words are then loaded into SR1, and a lookup is initiated. A compare command then causes the words within the selected dictionary segment to cycle through SR4, SR2, and SR3. SR2 accepts the dictionary words and moves them to the compare position (lines up beginning and end-of-word marks). SR3 receives the dictionary words from SR2. If a match does not occur, SR3 is cleared. If a match is found, the dictionary word is advanced to the end of SR3 and awaits call to the computer. If no match is found, the data word is considered not to be in the dictionary and thus not correctable.

For all other data word types, the threshold is initially set at one. In these cases, if no match is found, the threshold is increased by one and the lookup process is repeated. This cycle is repeated until a match is found or the threshold setting exceeds an arbitrary limit set for the various character lengths. The present formula for determining threshold limits is the integer portion of N/2, where N is the character length of the item. These figures have been previously determined and stored in a table. This table can be readily changed if future statistics prove the need to do so.

Once the lookup process on an item is completed, the item is flagged as corrected or not corrected along with the highest threshold value reached in the process. The flags are stored in the high order five bits of the begin word mark (174_8). If the word is considered to be corrected, bit 11 is a zero and bits 10-7 contain the binary form of the highest threshold setting used in correcting the data word. If the word is considered to be not corrected, then bit 11 contains a one and bits 10-7 contain the binary form of the arbitrary threshold setting. Items which are not alphabetic or those having character counts outside SRC limits are flagged as uncorrected but contain a zero in the threshold setting bits.

Another major part of the lookup process is selecting the proper segment of the dictionary to cycle over in attempting a match. Each track contains up to 814_{10} characters. Requirements for the selection process are a BANK and TRACK address and a WORD FROM and WORD TO address. The BANK and TRACK addresses are compiled and stored in a table when the dictionary is being loaded onto the disk. The table contains the BANK and TRACK address of the first word of each character length.

With this information, the program selects the proper BANK and TRACK address and gives an initial WORD FROM and WORD TO address ranging from 0 to 814_{10}. If no match is found in this segment, the BANK and TRACK address is increased by one. At this point a check is made to ensure that the proper character length is still being searched.

Owing to the alphanumeric sort order, the following feature allows for faster cycle time over selected positions of the disk. If a match is found in a particular segment, part of the return information is the address on the track where the match occurred. This address minus 20_{10} is placed in the WORD

FROM address in order to allow for the next items being duplicates of the item just matched (20 characters is the largest possible word i.e., 18 characters plus 2 word marks).

Once a lookup process has been initiated, the return information is checked to determine the result of the lookup. A return of zero means that a match has been found according to the conditions that were previously set. The dictionary word involved in the match is retrieved from SR3 to replace its counterpart in the data stream. In this manner all confusion and best guess characters are deleted and the word is considered to have been corrected. A return of five means the word has been passed and a match, under the set tolerance, was not found. This condition, performed by SRC hardware, is based on the sum of the first three characters of the data word being greater than the sum of the first three characters of the dictionary word. A return of seven means the SRC has cycled over the complete segment two times and has not found a match or passed beyond the data word alphanumerically. This is brought about by the hardware feature of stopping the cycle procedure when the Universal code 173_8 has been encountered twice. To make use of this feature, when the dictionary is being loaded, the code 173_8 is inserted as the last character in each track.

The general flow of this routine stems from the procedures set forth in ECP-2 Technical Note 56 and ECP-2 Technical Note 56, Supplement 1. The procedures described in these notes allow a program written in RADCAP to make use of the SRC and the BD-500 disk. Copies of these notes will be found in the appendixes of this report.

4.6.4 Output

The output of this routine is magnetic tape containing the input tape information with corrected data words and flags substituted for incorrect data words. Data words which remained uncorrected are composed of the original input data word with program inserted flags. Data words considered corrected are dictionary words that have replaced the original data word. Corrected data words also contain program inserted flags. Program inserted flags are explained in section 4.6.3. Appendix E contains examples of flag setup. This tape is used in the normal system flow as input to the RE-SORT routine. However, it can also be used as direct input to the STATISTICS I routine, since resorting is not necessary to gather the statistics on the correction procedures.

4.7 RE-SORT ROUTINE

4.7.1 Purpose

The RE-SORT routine (see Fig. 4-7) takes data which has been processed by the Spelling Correction routines, and which was previously sorted by character length and alphanumerically, and resorts it into its original order according to its document and item number.

4.7.2 Function

Blocks of data are read from magnetic tape, reordered by document and item

number, and written on another magnetic tape. Each block as it is sorted is merged with previously processed blocks until the entire input tape is resorted.

4.7.3 Input

The input for this routine is a magnetic tape containing blocks (512_{10}) of corrected data output from the SRC or SHORTWORD routines. Each batch of data (see Appendix B for an explanation of batching) has been sorted universally. The end of the input data is denoted by a double end-of-file block.

4.7.4 Method

In the following explanation, the notation MT1 denotes the input magnetic tape, MT2 and MT3 denote the intermediate merged tapes, and MTF denotes the final merged tape. MT2 and MT3 are logically switched at the end of each block merging operation. The last one used is written onto MTF when an end-of-file block is seen.

Three blocks of 512 locations each are used as memory storage and work areas. These areas are labeled NSB (newly sorted block), FSB (formerly sorted block), and NMB (newly merged block) for brevity.

One block of data is read into NSB from MT1 and is sorted by document number. This is done by repeated scans of NSB, looking for the lowest document number seen on each scan as each item of NSB is examined. Whenever a new low document number is seen, this number replaces the old low document number and the item itself is copied into FSB, replacing any former low item, until at the completion of one scan the lowest document numbered item has been transferred. Each transferred item is "erased" from NSB by zeroing its "key" register (174) so that it is ignored on the next scan. Thus, one item is transferred to FSB at each scan, while the scans become progressively shorter until all items have been transferred and arranged in ascending document number order.

The same process is repeated with the item numbers of each document being arranged numerically as they are transferred back out of FSB into NSB. At the completion of this transfer, the block of data is fully resorted and ready to be merged with previously processed data.

In the case of the very first block, no previously sorted data yet exists, so the block is immediately written onto MT2. A new block is then read from MT1 to NSB and sorted by document and item number. The previously sorted block is read from MT2 into the FSB area. Item 1 of NSB is compared to item 1 of FSB and the lower numbered item is transferred to NMB. In this fashion, the items of both FSB and NSB are merged and moved to NMB until NMB is within 56 registers of being full. NMB is then written onto MT3, where it may be used in the merging of the next input block. The remaining items of FSB and NSB are merged as NMB is refilled.

When FSB is emptied, another block is read into it from MT2 and merging continues. When NSB is emptied, the program is adjusted to make an NSB entry appear to have a value so high that all remaining items from MT2 are selected as lower than the NSB item. When MT2 is exhausted of items and NSB is also

(a) Sort

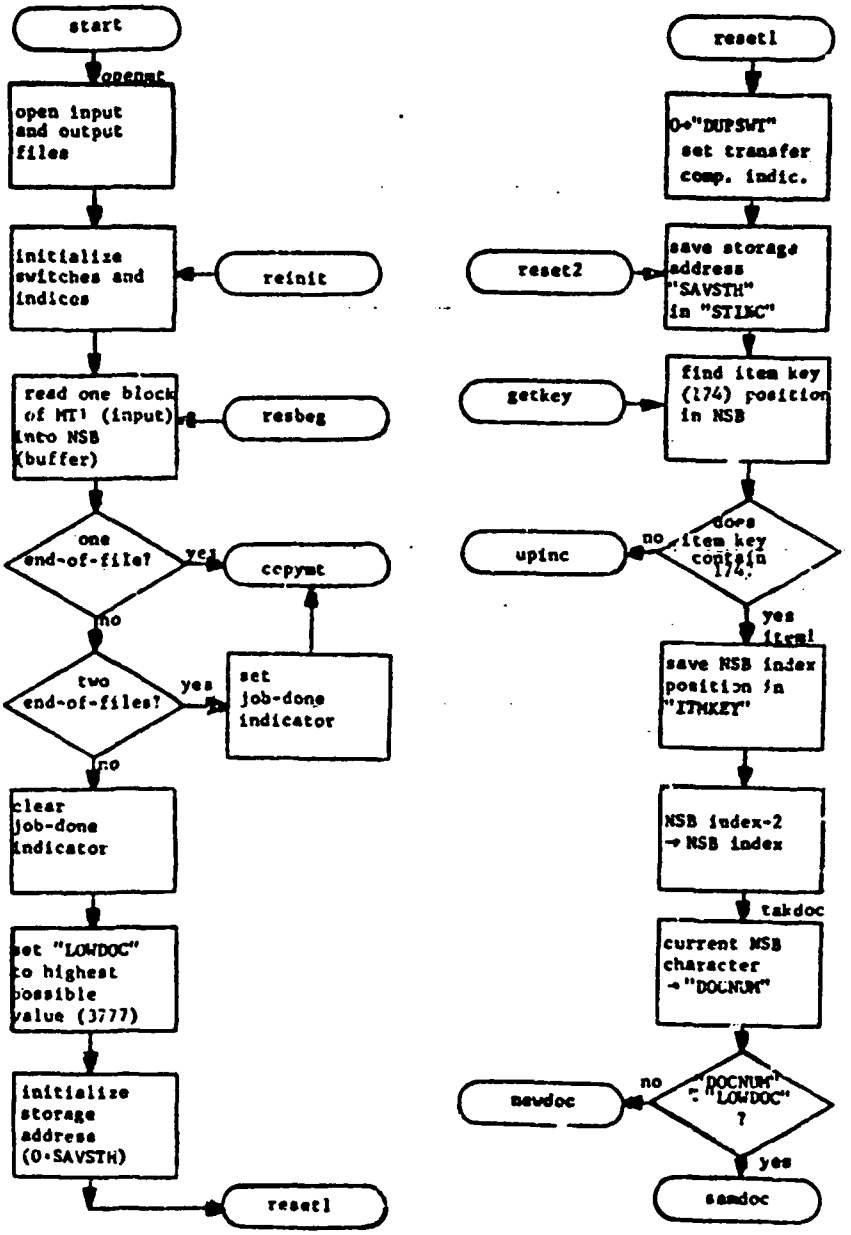


Fig. 4 - 7 Flow diagram for Re-Sort routine

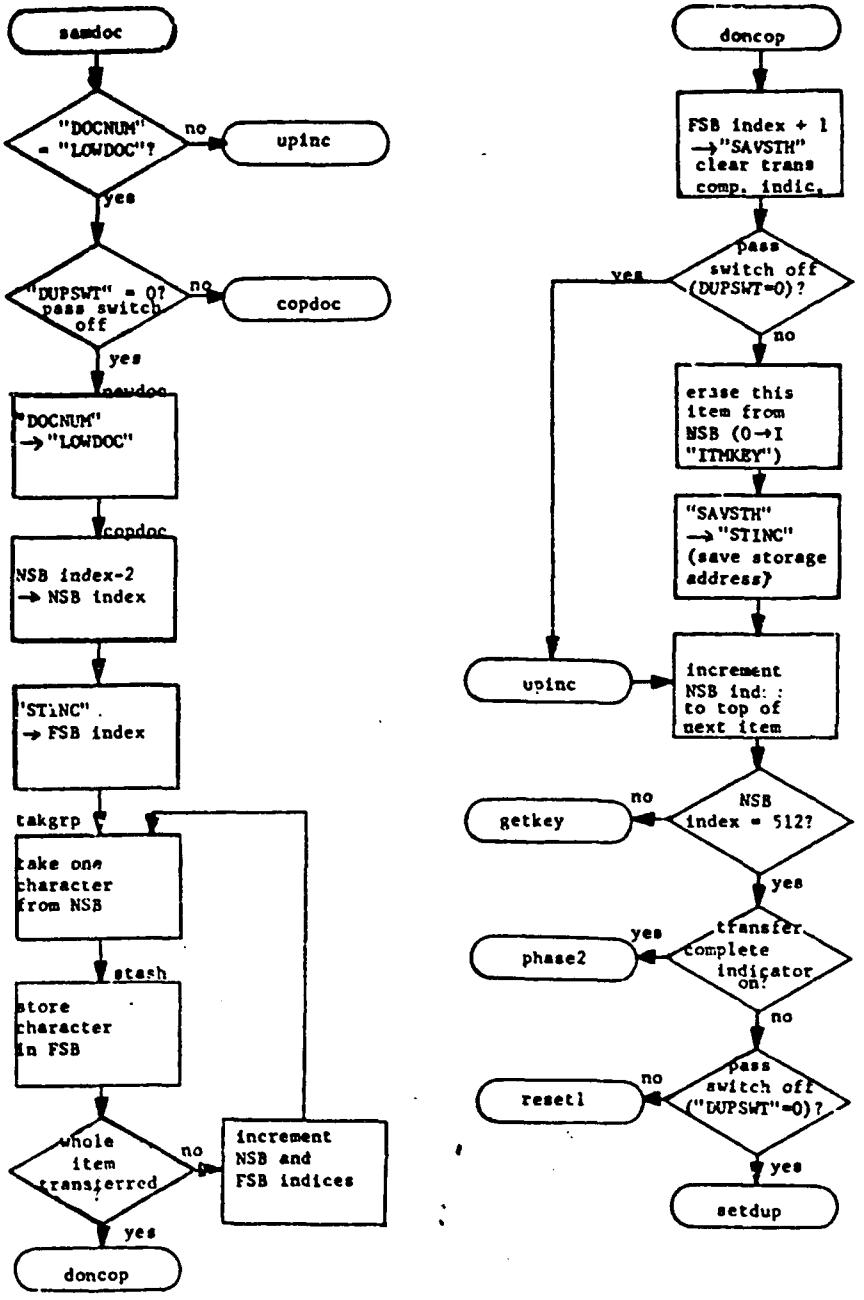

samdoc
"DOCNUM" = "LOWDOC"?
no
upinc
yes
"DUPSWT" = 0? pass switch off
no
copdoc
yes
newdoc
"DOCNUM" → "LOWDOC"
copdoc
NSB index-2 → NSB index
"STINC" → FSB index
takgrp
take one character from NSB
stash
store character in FSB
whole item transferred
no
increment NSB and FSB indices
yes
doncop
doncop
FSB index + 1 → "SAVSTH" clear trans comp. indic.
pass switch off (DUPSWT=0)?
yes
no
erase this item from NSB (0→I "ITMKEY")
"SAVSTH" → "STINC" (save storage address)
upinc
increment NSB ind: : to top of next item
NSB index = 512?
no
getkey
yes
transfer complete indicator on?
yes
phase2
no
pass switch off ("DUPSWT"=0)?
no
reset1
yes
setdup

Fig. 4 - 7 (cont'd)

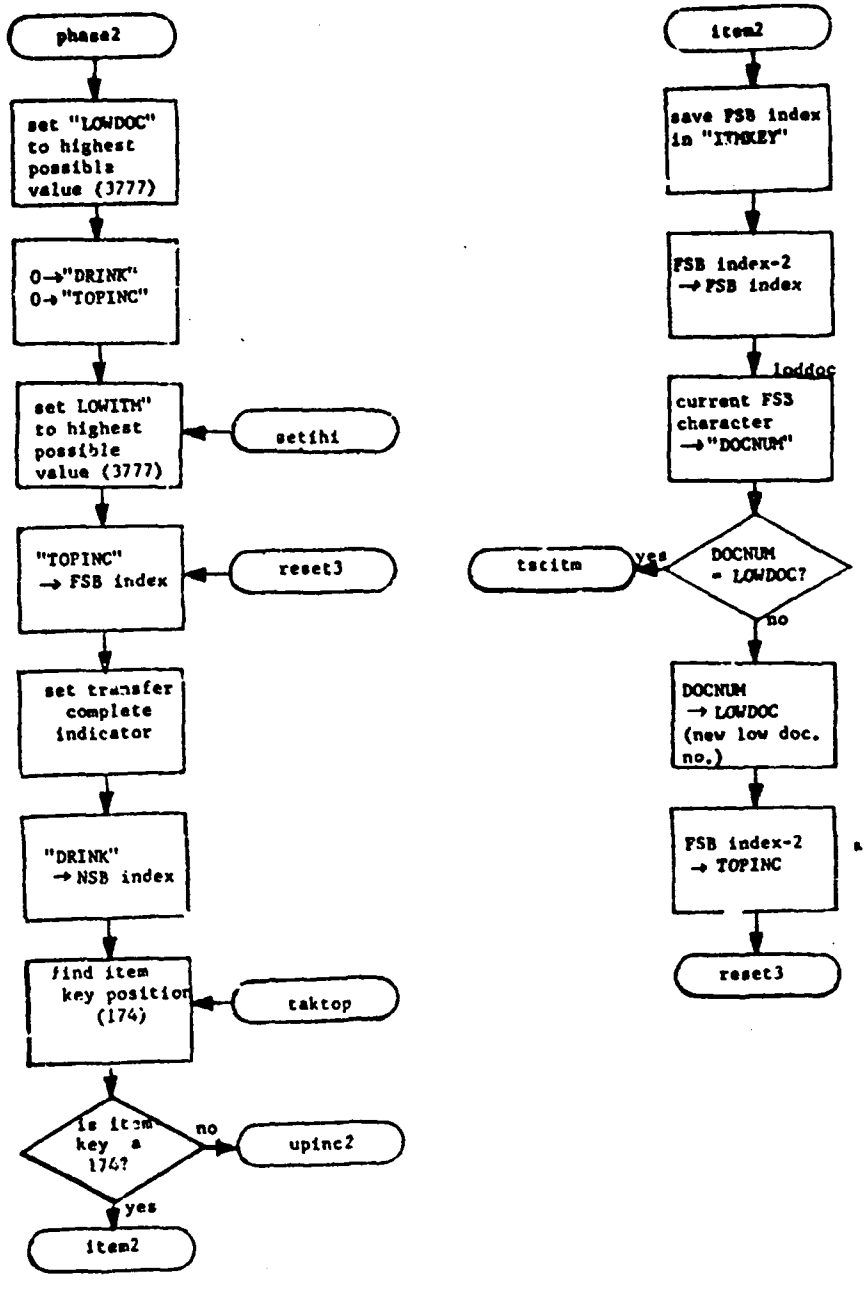

phase2
set "LOWDOC" to highest possible value (3777)
0→"DRINK"
0→"TOPINC"
set LOWITM" to highest possible value (3777)
setihi
"TOPINC" → FSB index
reset3
set transfer complete indicator
"DRINK" → NSB index
find item key position (174)
taktop
is item key a 174?
no
upinc2
yes
item2
item2
save FSB index in "ITMKEY"
FSB index-2 → FSB index
ioddoc
current FSS character →"DOCNUM"
tscitm
yes
DOCNUM = LOWDOC?
no
DOCNUM → LOWDOC (new low doc. no.)
FSB index-2 → TOPINC
reset3

Fig. 4 - 7 (cont'd)

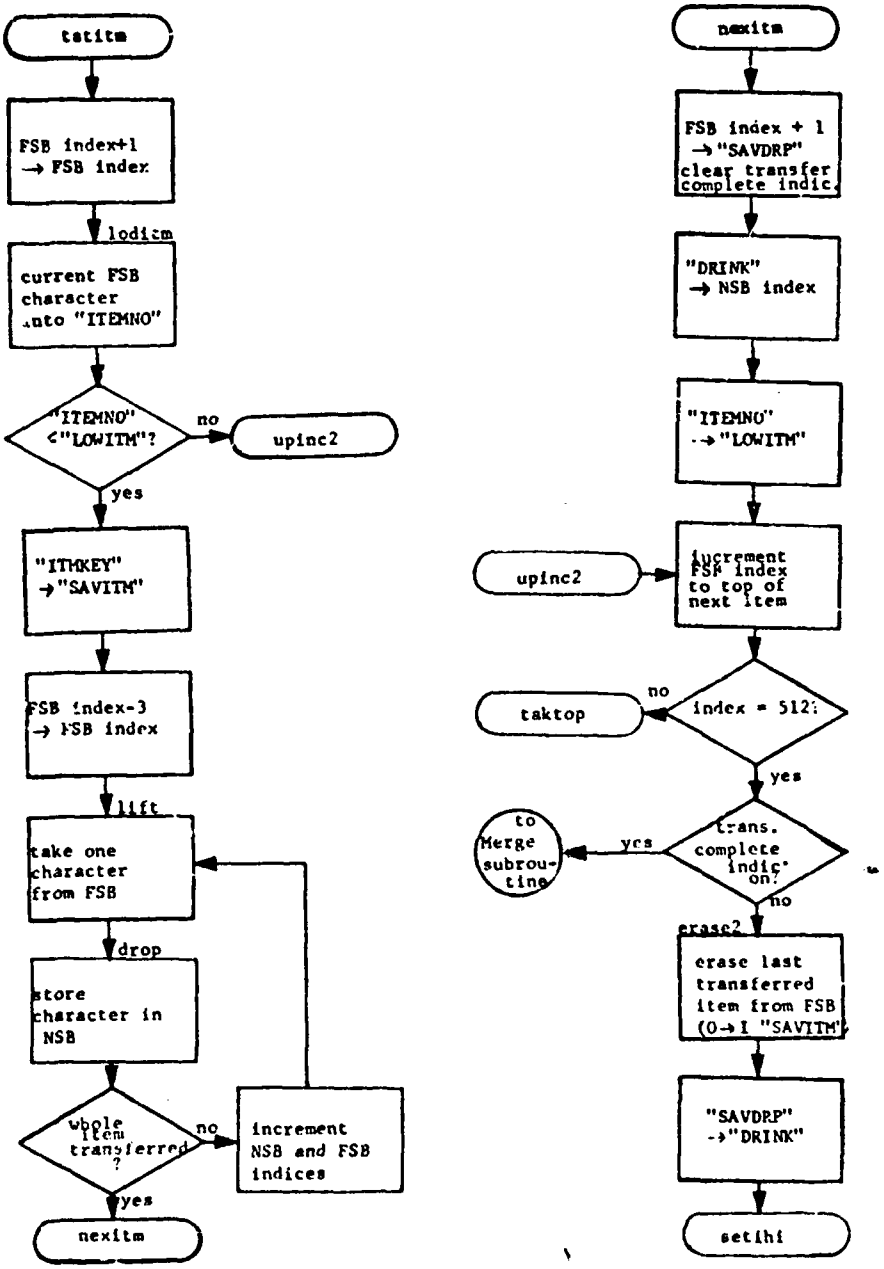

tatitm
FSB index+1
→ FSB index
lodicm
current FSB
character
into "ITEMNO"
"ITEMNO"
<"LOWITM"?
no
upinc2
yes
"ITMKEY"
→"SAVITM"
FSB index-3
→ FSB index
lift
take one
character
from FSB
drop
store
character in
NSB
whole
item
transferred
?
no
increment
NSB and FSB
indices
yes
nexitm
nexitm
FSB index + 1
→"SAVDRP"
clear transfer
complete indic.
"DRINK"
→ NSB index
"ITEMNO"
→"LOWITM"
upinc2
increment
FSB index
to top of
next item
taktop
no
index = 512?
yes
to
Merge
subrou-
tine
yes
trans.
complete
indic
on?
no
erase2
erase last
transferred
item from FSB
(0→I "SAVITM"
"SAVDRP"
→"DRINK"
setihi

Fig. 4 - 7 (cont'd)

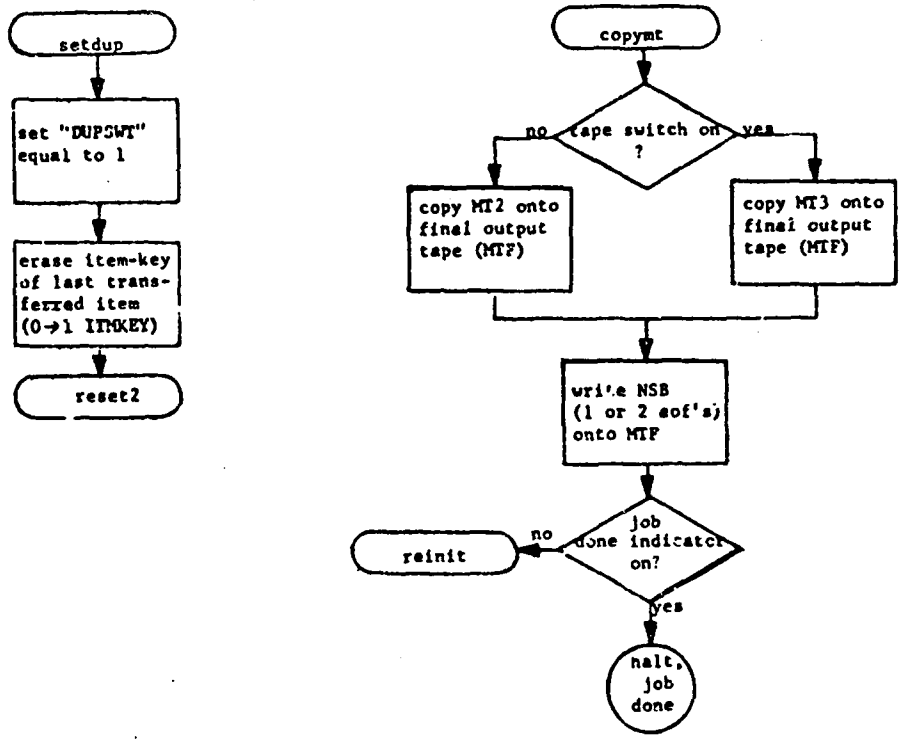

setdup
set "DUPSWT" equal to 1
erase item-key of last trans-ferred item (0→1 ITMKEY)
reset2
copymt
no
tape switch on ?
yes
copy MT2 onto final output tape (MTF)
copy MT3 onto final output tape (MTF)
write NSB (1 or 2 eof's) onto MTF
job done indicator on?
no
reinit
yes
halt, job done

Fig. 4 - 7 (cont'd)

(b) Merge

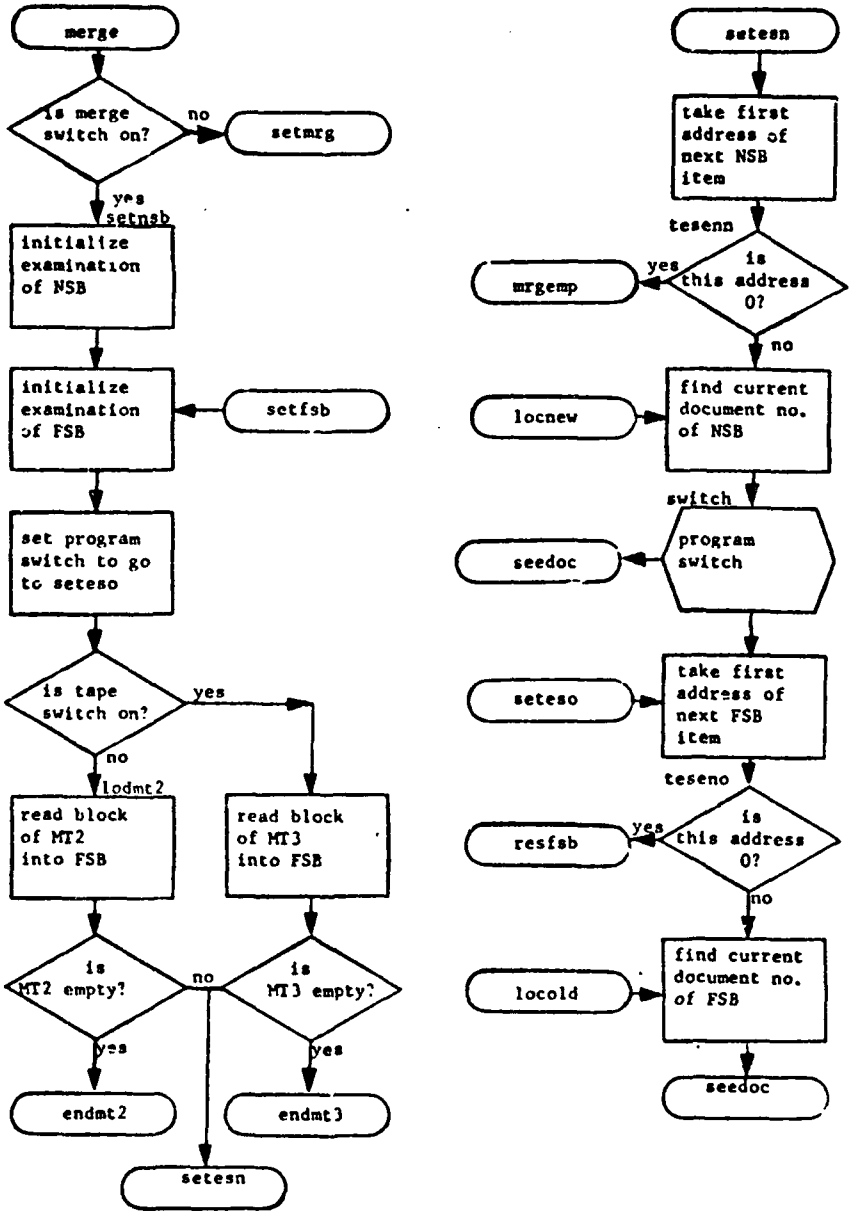


Fig. 4 - 7 (cont'd)

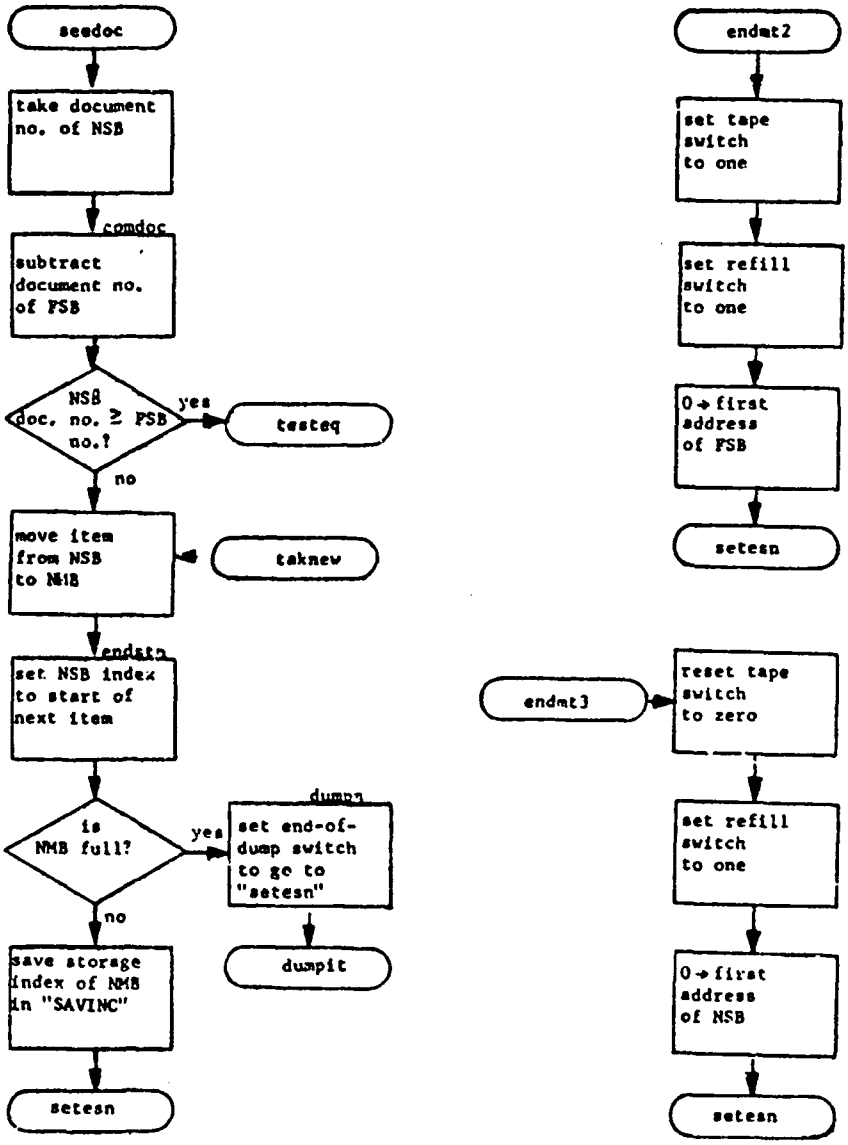
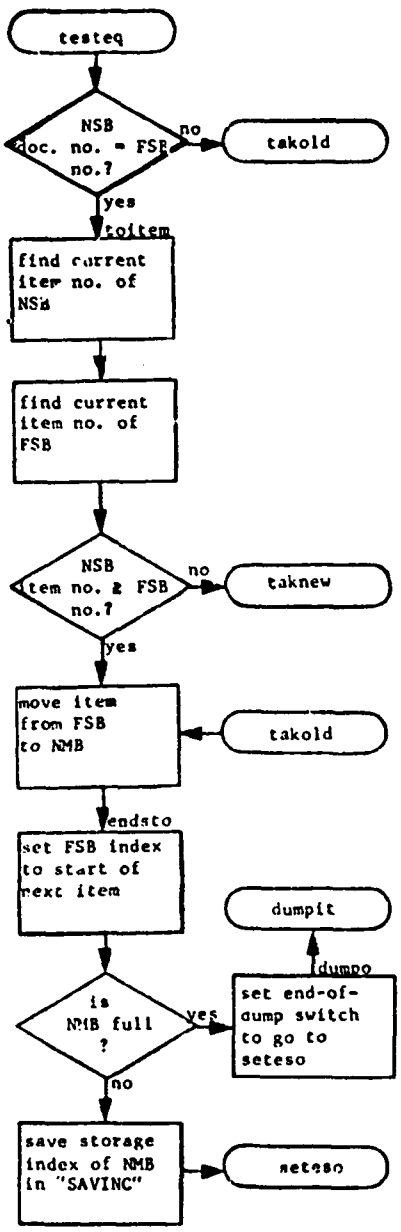

seedoc
take document no. of NSB
comdoc
subtract document no. of FSB
NSB doc. no. ≥ FSB no.?
yes
testeq
no
move item from NSB to NMB
taknew
endstn
set NSB index to start of next item
is NMB full?
yes
dumpn
set end-of-dump switch to go to "setesn"
dumpit
no
save storage index of NMB in "SAVINC"
setesn
endmt2
set tape switch to one
set refill switch to one
0 → first address of FSB
setesn
endmt3
reset tape switch to zero
set refill switch to one
0 → first address of NSB
setesn

Fig. 4 - 7 (cont'd)


testeq
NSB doc. no. = FSB no.?
no
takold
yes
toitem
find current item no. of NSB
find current item no. of FSB
NSB item no. ≥ FSB no.?
no
taknew
yes
move item from FSB to NMB
takold
endsto
set FSB index to start of next item
dumpit
dumpo
is NMB full ?
yes
set end-of-dump switch to go to seteso
no
save storage index of NMB in "SAVINC"
seteso

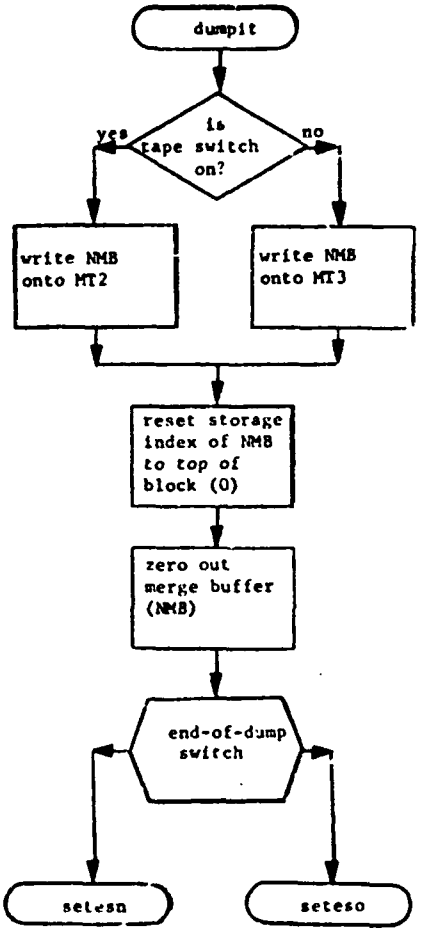

dumpit
is tape switch on?
yes
no
write NMB onto MT2
write NMB onto MT3
reset storage index of NMB to top of block (0)
zero out merge buffer (NMB)
end-of-dump switch
setesn
seteso

Fig. 4 - 7 (cont'd)

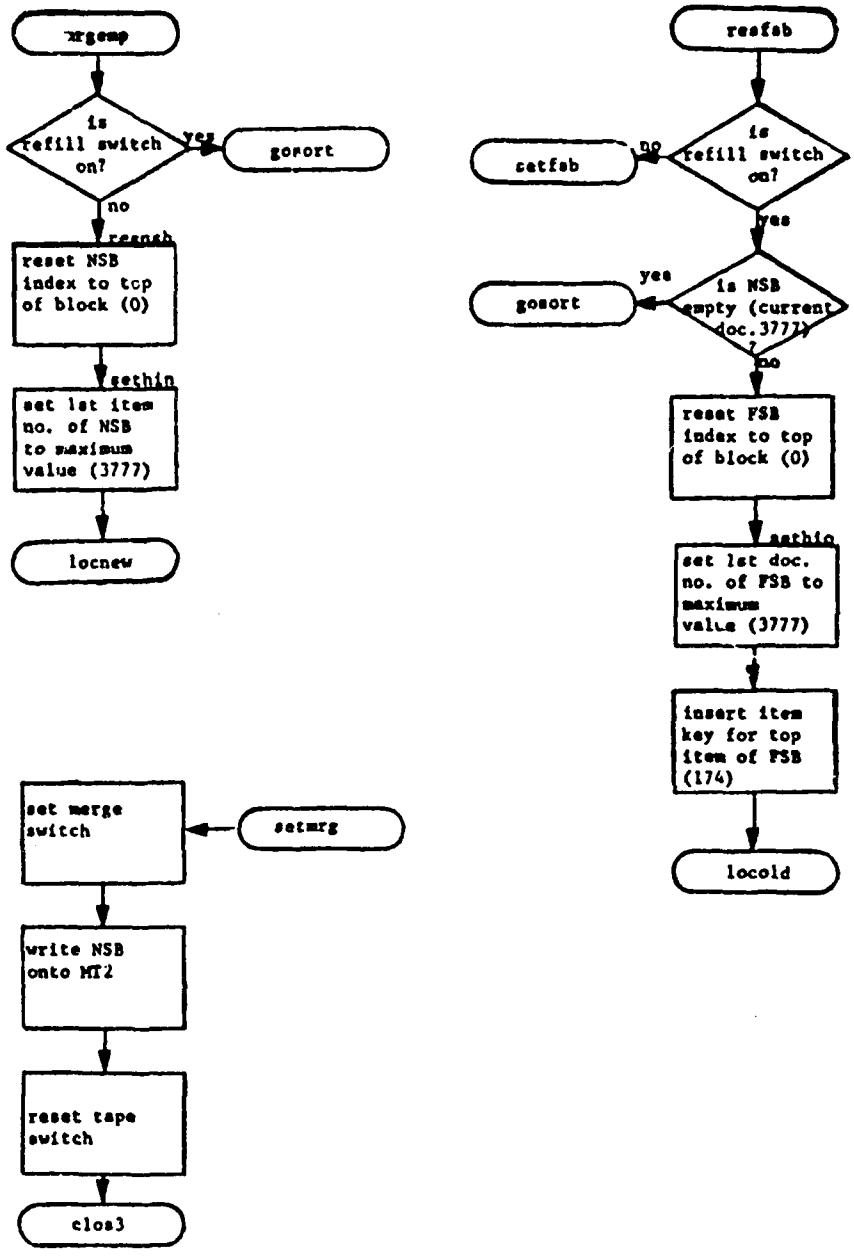

xrgcmp
is refill switch on?
yes
gosort
no
reqnsb
reset NSB index to top of block (0)
setbin
set lst item no. of NSB to maximum value (3777)
locnew
resfsb
is refill switch on?
no
setfsb
yes
is NSB empty (current doc.3777)
yes
gosort
no
reset FSB index to top of block (0)
setbig
set lst doc. no. of FSB to maximum value (3777)
insert item key for top item of FSB (174)
locold
set merge switch
setmrg
write NSB onto MT2
reset tape switch
clos3

Fig. 4 - 7 (cont'd)

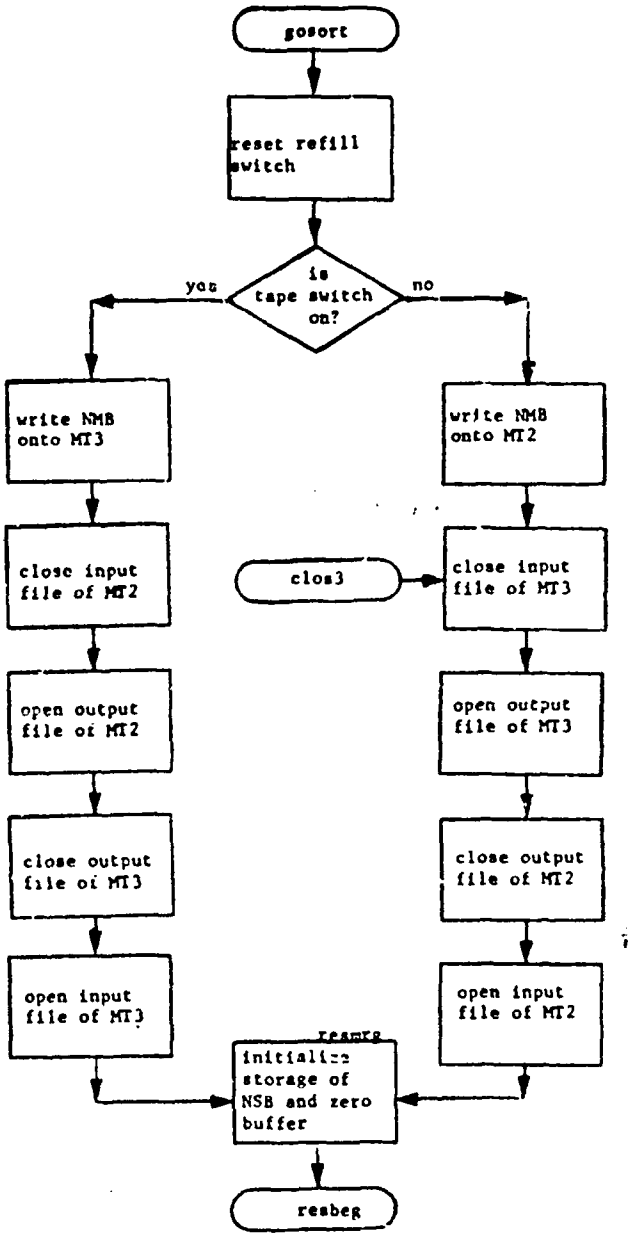

gosort
reset refill switch
is tape switch on?
yes
no
write NMB onto MT3
write NMB onto MT2
close input file of MT2
clos3
close input file of MT3
open output file of MT2
open output file of MT3
close output file of MT3
close output file of MT2
open input file of MT3
open input file of MT2
resmrg
initialize storage of NSB and zero buffer
resbeg

Fig. 4 - 7 (cont'd)

empty, MT3 becomes the merged tape. It is used to merge with another input block (if there is one), and the new results are written onto MT2. So long as there is new input to read from MT1, this process continues with MT2 and MT3 alternately becoming the output tape.

A single end-of-file block in the input tape indicates the end of a batch. This causes the current output tape (MT2 or MT3) to be copied onto the final output tape before the next block is read from MT1. A double end-of-file block on the input tape indicates the end of the job. Again, the current output tape is copied onto the final output tape (along with the double end-of-file block) before the routine halts.

4.7.5 Output

The output tape for this routine is a magnetic tape containing the corrected version of the data words in their original print reader order. However, it is not in the same format as the PEELER routine input, since the six descriptive words added by the PEELER routine and the batching scheme added by the SORT routine are still present. This tape is normally used as input to the DISPLAY routine but can also be used as input to both STATISTICS routines.

4.8 STATISTICS I (WORST) ROUTINE

4.8.1 Purpose

The purpose of the STATISTICS I routine (see Fig. 4-3) is to gather statistics concerning alphabetic words of 18 or less characters. The statistics are compiled separately for each character length. Within each length, statistics for frequency count and the occurrence of best guess, confusion, confusion and best guess, corrected, and uncorrected words are kept.

4.8.2 Input

The input tape can be the output tape of either the SRC routine or the RE-SORT routine. Although it is necessary to input all words of an item, only three words are of direct concern to this routine. They are the first, second, and fifth words of an item; the character count, type code, and begin word, respectively.

4.8.3 Description

After the file has been opened, the output headings are set up and all storage areas are cleared. Next a block of data is read. If it begins with either a single or double end-of-file block, that batch is displayed. The display section of this routine is explained later.

If no display took place, the character count is examined and if it is within the limits for the length of a word (i.e., $1 \leq n \leq 18$, where n is the character count), then a tally is made in the appropriate frequency count. Separate tallies are kept for each length. If the length is above the limit, the character count of the next item is examined. If it is below, the next block is read.

Next the type code is examined. If the word is not alphabetic or is hyphenated, the next item is examined. If the word is purely alphabetic and is unhyphenated, a tally is made in one of the three classes: best guess, confusion, or both. The class in which to make the tally is determined by the arrangement of the four leftmost significant bits of the type code word:

0000	alphabetic characters only
0100	alphabetic characters with confusion
0010	alphabetic characters with best guess
0110	alphabetic characters with both

After this, the begin word is examined. If there is a one in bit 11, then it was not possible to correct the word, so a tally is made in the uncorrected total, or in the corrected total if bit 11 was zero. Then the next item is examined until the input block is completely processed, in which case the next block is read from the input tape.

If it had been necessary to output a block, since an end-of-file block was encountered, then the batch number would have been displayed first. Then the headings, which were set up at the start of the routine, are displayed, followed below by the appropriate statistics. Before the statistics are displayed they are converted from octal to decimal form. When all the statistics for this batch have been displayed, the next block of data is read, unless there has been a double end-of-file block. A double end-of-file block terminates this routine.

4.8.4 Output

The output is displayed on the typewriter. All data within a set of end-of-file blocks is considered a batch. Whenever an end-of-file block is read, all the statistics obtained for that batch are displayed. First, the batch number is typed, followed by the headings. The headings indicate the various categories: character length, frequency, best guess, confusion, confusion and best guess, corrected, and uncorrected. Then under each of the seven categories the proper statistics are typed. This typing of batch number, headings, and statistics occurs after each batch has been processed (see Appendix F for a sample output).

4.9 STATISTICS II (PONCW) ROUTIN

4.9.1 Purpose

The purpose of the STATISTICS II routine (see Fig. 4-9) is to obtain a printout of all alphabetic items that were not corrected by the programming logic in the SRC or SHORTWORD routines. The printout of each item is preceded by the maximum tolerance which was used in the correction attempt. Any confusion characters within an item are printed out as asterisks.

4.9.2 Input

The input tape for this routine is the output tape of the SRC or SHORTWORD routine. The input tape is read in blocks of 512 computer words. Within each of these blocks there is a variable number of items. Only complete items are placed in a data block, and unused locations, if any, are filled with zeros.

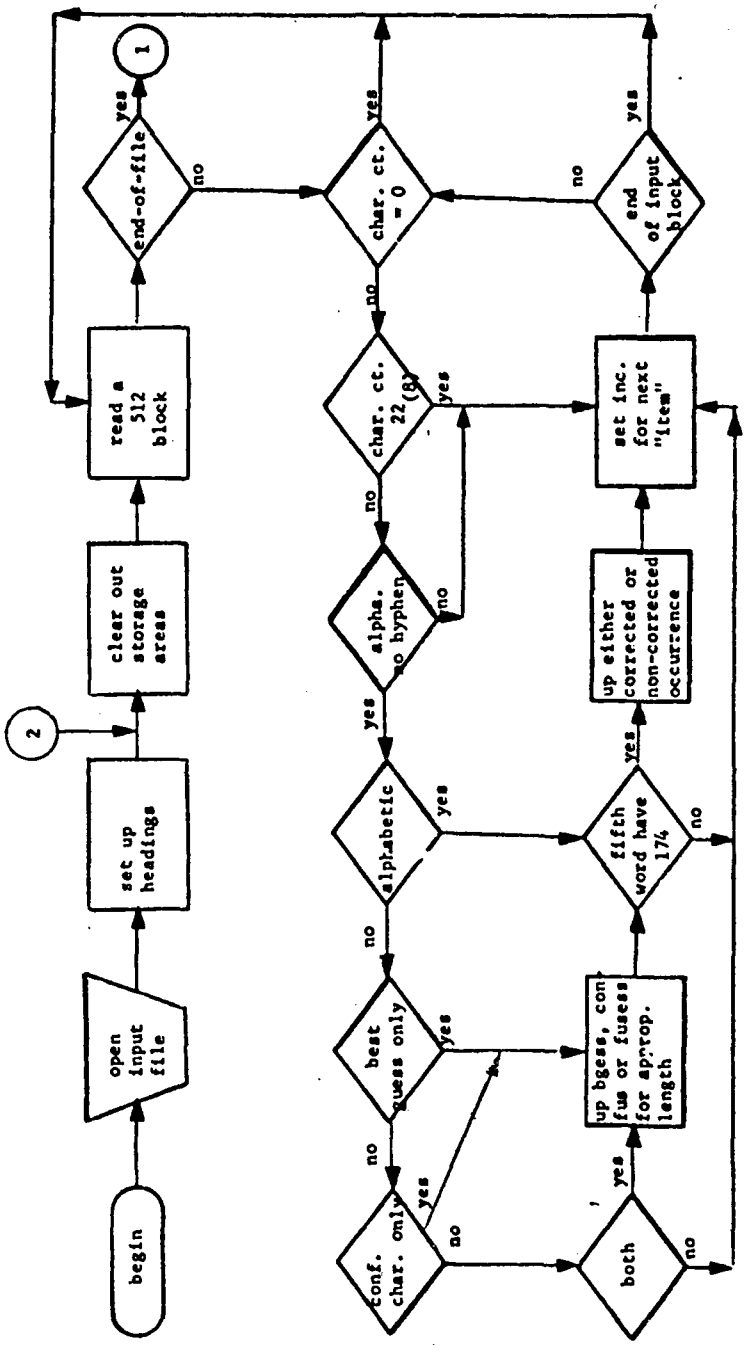


Fig. 4 - 8 Flow diagram for Dictionary Word Statistics routine

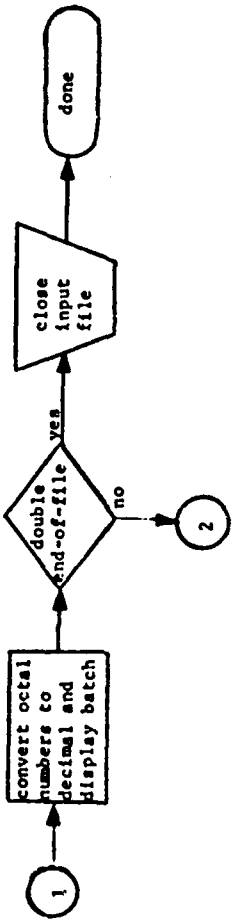

done
close input file
yes
double end-of-file
no
2
convert octal numbers to decimal and display batch
1

Fig. 4 - 8 (cont'd)

The following diagram illustrates the composition of an item and those words which are examined by this routine.

Word	Description	Used or Not Used
1	Number of characters in item	Used
2	Type code	Not used
3	Document number	Not used
4	Item position in document	Not used
5	174_8 and tolerance level	Only tolerance level is used
.	I	Used
.	T	Used
.	E	Used
.	M	Used
n	175_8	Not used

4.9.3 Description

After an input block has been read into memory, a check is made to see if it is a single or double end-of-file block. A single end-of-file block is ignored and the next block on the input tape is read in. A double end-of-file block indicates the end of the input data and the contents of the output buffer are printed out.

If no end-of-file block is encountered, the character count of each data word is examined for a zero. A zero character count implies that the 512-word data block has been completely processed and the next block is read into memory. If it is not zero, then the number four is added to the character count to make room for additional information needed in the output buffer. Bit 11 of the fifth word of each item is then examined to determine if the data word is uncorrected or corrected. For corrected words all but bits 7-10 of the fifth word are cleared and the routine moves on to the next data word.

The item tolerance level for all uncorrected data words is converted from binary to Universal code (only code accepted by the printer) and stored with the data in the output buffer. Any confusion characters that may be present in the data word are converted to asterisks for printing purposes. A carriage return is stored in the output buffer following each entry so the output will be in a single column.

A check is performed each time an entry is stored in the output buffer for the end of the buffer. When the end of the buffer is encountered, an end-of-message code is stored and the output buffer is printed.

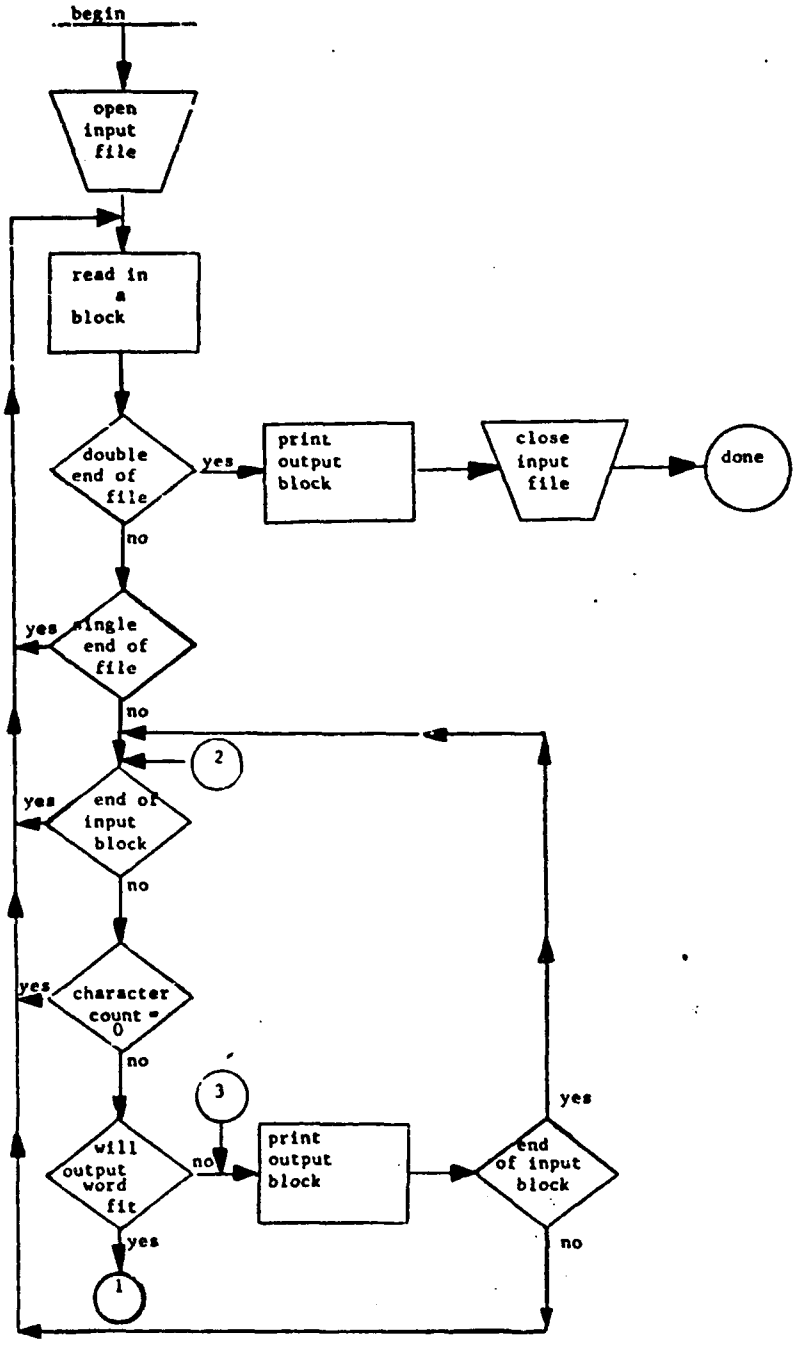


Fig. 4 - 9 Flow diagram for Non-Corrected Word routine

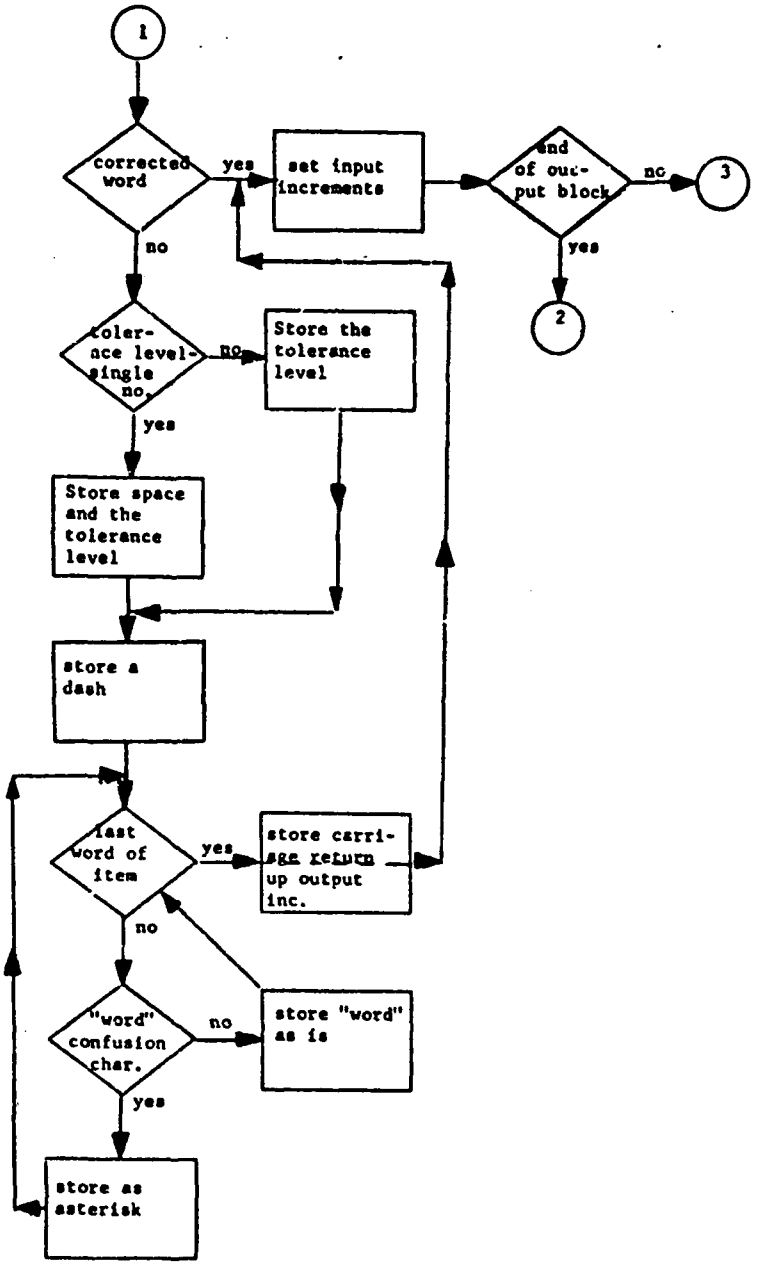

1
corrected word
yes
set input increments
end of output block
no
3
yes
2
no
tolerance level-single no.
no
Store the tolerance level
yes
Store space and the tolerance level
store a dash
last word of item
yes
store carriage return up output inc.
no
"word" confusion char.
no
store "word" as is
yes
store as asterisk

Fig. 4 - 9 (cont'd)

4.9.4 Output

Each time the 512-word output buffer is filled, it is printed. The tolerance level for uncorrected data words precedes the word itself as shown by the following example:

```
3  ABACUS
1  B*T
2  CH*I*
6  UNCONFORMABLE
```

Note that the words such as "abacus" and "unconformable" will appear on the printout as uncorrected words even when they are correct if they are not contained in the dictionary.

4.10 DISPLAY ROUTINE

4.10.1 Purpose

The purpose of the DISPLAY routine (see Fig. 4-10) is to enable the presentation of the corrected text on the 408-A Datacom. All lines of the text are indented nine spaces. If a line of text exceeds the length of the scope line, the remainder of the line is placed on the next line after an added indentation of five spaces. Thus, although the number of lines is not the same, the first word of each of the original lines is easily recognized.

4.10.2 Input

The input tape used in this routine is the output tape from the RE-SORT routine. It is entered in blocks of 512 computer words. All words of an item are entered, although not all of them are used. The order of the words in an item and the fact of whether they are used or not is shown in the following table:

Word	Description	Used or Not Used
1	Number of characters in word	Used
2	Type code	Not used
3	Document number	Not used
4	Item position in document	Not used
5	174_8	Not used
.	I	Used
.	T	Used
.	E	Used
.	M	Used
n	175_8	Not used

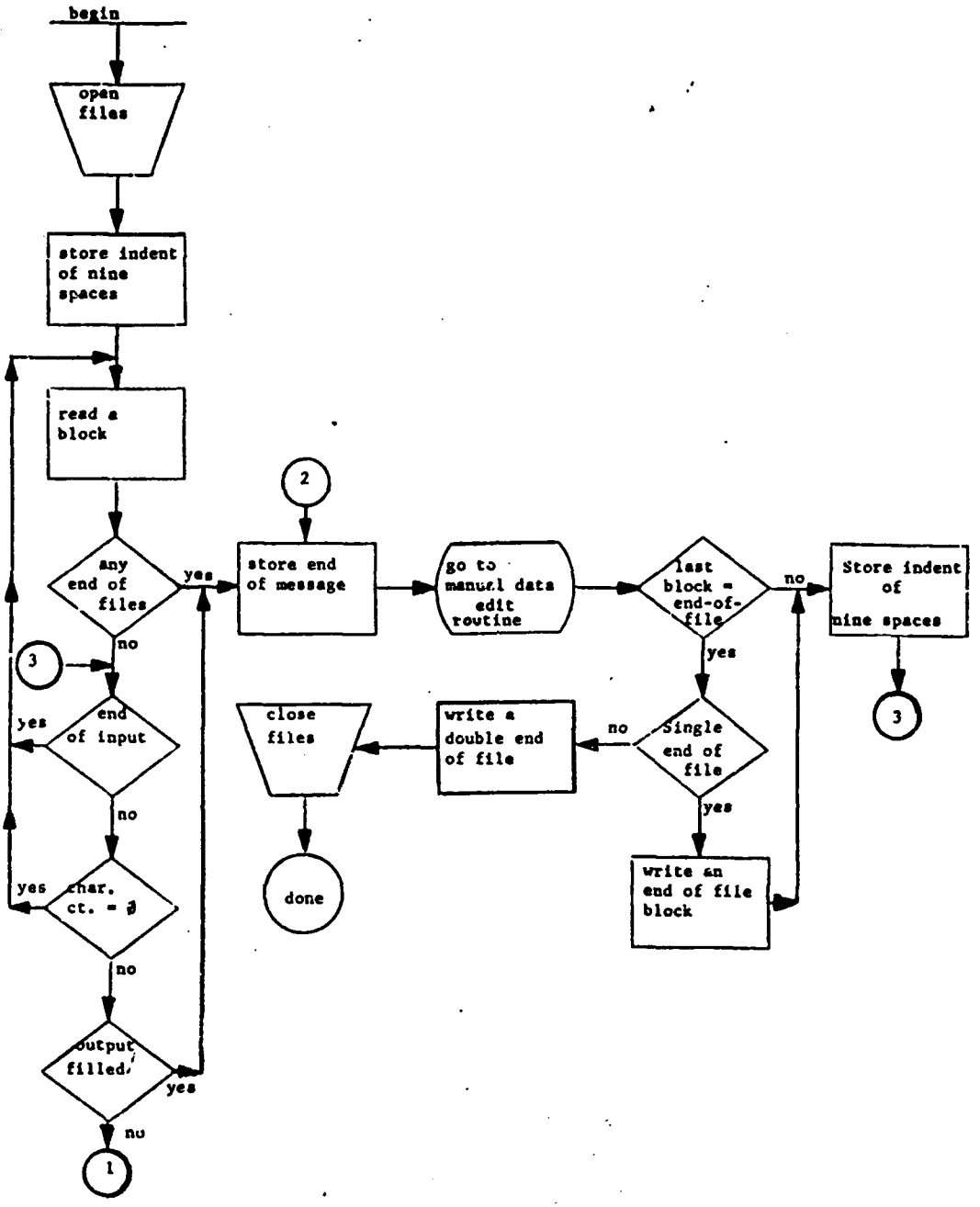
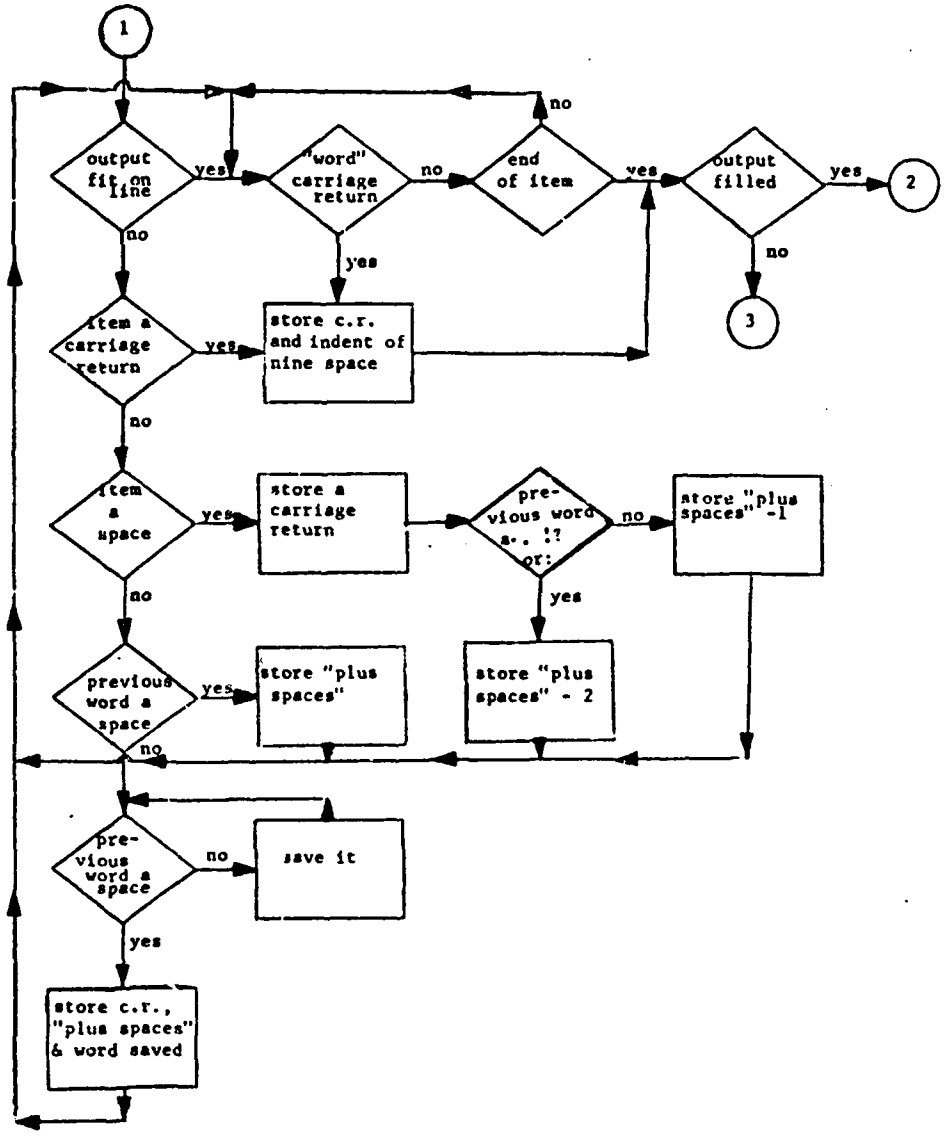


Fig. 4 - 10 Flow diagram for Display routine



Note: The term "plus spaces" means spaces to end of line plus 14 more

Fig. 4 - 10 (cont'd)

4.10.3 Description

After each input block is read, a check for an end-of-file block is made. If either a single (zeof) or a double (zeofzeof) end-of-file block is read, then the contents of the output buffer are displayed, written on the output tape, and the corresponding end-of-file block is also written. If a double end-of-file block is read, the routine terminates, but if only a single end-of-file block is read, the next block is brought in from the input tape.

If no end-of-file block is encountered, the input increment is tested. If it is at 512, a new block is read; otherwise, the first word of the item is examined. If it is zero, a new input block is read; otherwise the length of the output item is added to the output increment to see if the item will fit. If it will not fit, the output block is displayed; otherwise the length is checked to see if it will fit on the scope line, which is 63 characters long. If the item fits, it is stored and a check is made for a carriage return. If there is none, the routine proceeds with the next item. If there is one, the remainder of the line is filled with spaces, the next line is indented nine spaces, and the next item is processed.

If the item will not fit on the line, it is necessary to add a carriage return and enough spaces so that the remainder of the scope line is filled. Before the carriage return is inserted, a check is made to determine how many spaces must be added so that the indentation on the next line is 14 spaces. If the next item is a space, 13 spaces are to be inserted, unless the previous item is a question mark, period, colon, or exclamation point; then only 12 spaces are needed. If the next item is not a space, but the previous word was, then 14 spaces are inserted. Otherwise the previous words are searched until a space is found; then 14 spaces are added and the previous word or words are stored after the indentation. Then the program proceeds with the next item.

4.10.4 Output

After a 512-word output block has been filled, it is displayed on the scope. The maximum length of any line is 63 characters. Thus, if the original text was as follows:

> A study is made of the temperature field before the combustion
> front in a pipe with circular cross section. In the study
> it is assumed that the air moving along the pipe is heated on account
> of the stationary process of convective head exchange from the pipe.

the scope output would look like this:

```
A study is made of the temperature field before the

     combustion

front in a pipe with circular cross section.  In the

     study
```

it is assumed that the air moving along the pipe is

heated on account

of the stationary process of convective heat exchange

from the pipe.

4.11 MANUAL DATA EDIT (MDE) ROUTINE

4.11.1 Purpose

The purpose of the MANUAL DATA EDIT routine (see Fig. 4-11) is to allow an operator to remove any remaining errors left by the automatic correction procedures or to perform a general data editing function.

4.11.2 Input

The input to this routine is the displayed output of the DISPLAY routine. Each time the DISPLAY routine displays a block of corrected data it calls this routine. Other input is provided by the operator who is manning the scope. This includes names of subroutines which he selects to assist in the manual processing of the displayed data.

4.11.3 Function

The DISPLAY routine displays the automatically corrected text on the 408-A Datacom using a RADCAP KEY IN instruction. Control is then sent to the MANUAL DATA EDIT package, incorporated into the DISPLAY routine, to accept and interpret messages, if any, inserted by the operator. For the present, only a few routines are available to the operator, but the package is designed to permit additional functions to be included at a later time.

The present package includes a routine, SWB35, to switch the function of flag bits 3 and 5 of the displayed text. Using this routine, an operator can substitute the clearly visible overline marker on the Datacom for those characters which were flagged as best guess. Calling this routine again restores the display to its original format. A second routine called ERROR is available to reinitialize the original display should the operator have inadvertently destroyed some valuable data. A third routine, STOP, is available to act as a function delimiter for the other routines. Each of the routines operates on a line, or lines, of the displayed text. Thus a routine name in the left-hand display margin next to a particular line operates on that line and each succeeding line of the display until either the end of the display is reached or another routine name is found in the display margin.

To switch the function of flag bits 3 and 5 for only a single line in the display, one would type the routine name SWB35 in the display margin next to the line one wished to operate on and type STOP in the display margin next to the following line. Here the routine STOP serves as a delimiter for the SWB35 routine.

Each routine name must be seven characters or less in length and be followed by a blank. Imbedded blanks in the name are not permitted. Routine names do not have to be either left or right adjusted in the display margin.

If an operator should request a routine which is not available or not called correctly, the package removes the name from the display margin and continues operation.

After a routine has performed its intended function, it removes its own name from the display margin and returns control to the scanning section of the package to look for other routines which may be called. When the end of the display is reached and a routine has been called at some previous point, control is returned to the display routine to repeat this same display with the changes incorporated. In this manner an operator can perform many separate functions on a single display. When no further routines are requested, the last display is written on the final output tape and a new block is displayed.

4.11.4 Method

The display margin is scanned from top to bottom for nonblank characters. A nonblank character transfers control to a matching section which compares the characters found against a subroutine table. Each entry in the subroutine table is followed by a blank (012B) and an exact match is required with the subroutine table including the blank character.

The subroutine search is accomplished using an alphanumeric lookup technique where the names in the subroutine table must be ordered according to their Universal code values. This is the same order output by the SORT and MERGE routines for the document data.

When a legitimate subroutine name has been requested and a match found in the subroutine table, the package constructs a RADCAP JUMP instruction to a JUMP TABLE which sends control to the appropriate routine.

This description is concerned only with existing subroutines which assist the operator in correcting or editing the displayed data. There are of course many editing and correcting aids built into the 408-A Datacom hardware which increase the capability of an operator, but these are explained in the appropriate manuals for the 408-A Datacom.

4.11.5 Output

This routine does not write on an output device. The results of this routine are direct changes to the data held in the DISPLAY routine's output buffer.

Fig. 4-12a is an example of the lower half display of the Datacom console after an operator has partially corrected the text. In the left-hand margin, the operator has typed the name of a subroutine (SWB35) which he selected to use in further correcting the text. Subroutine names can be placed anywhere in the first eight positions of the margin but imbedded blanks are not permitted.

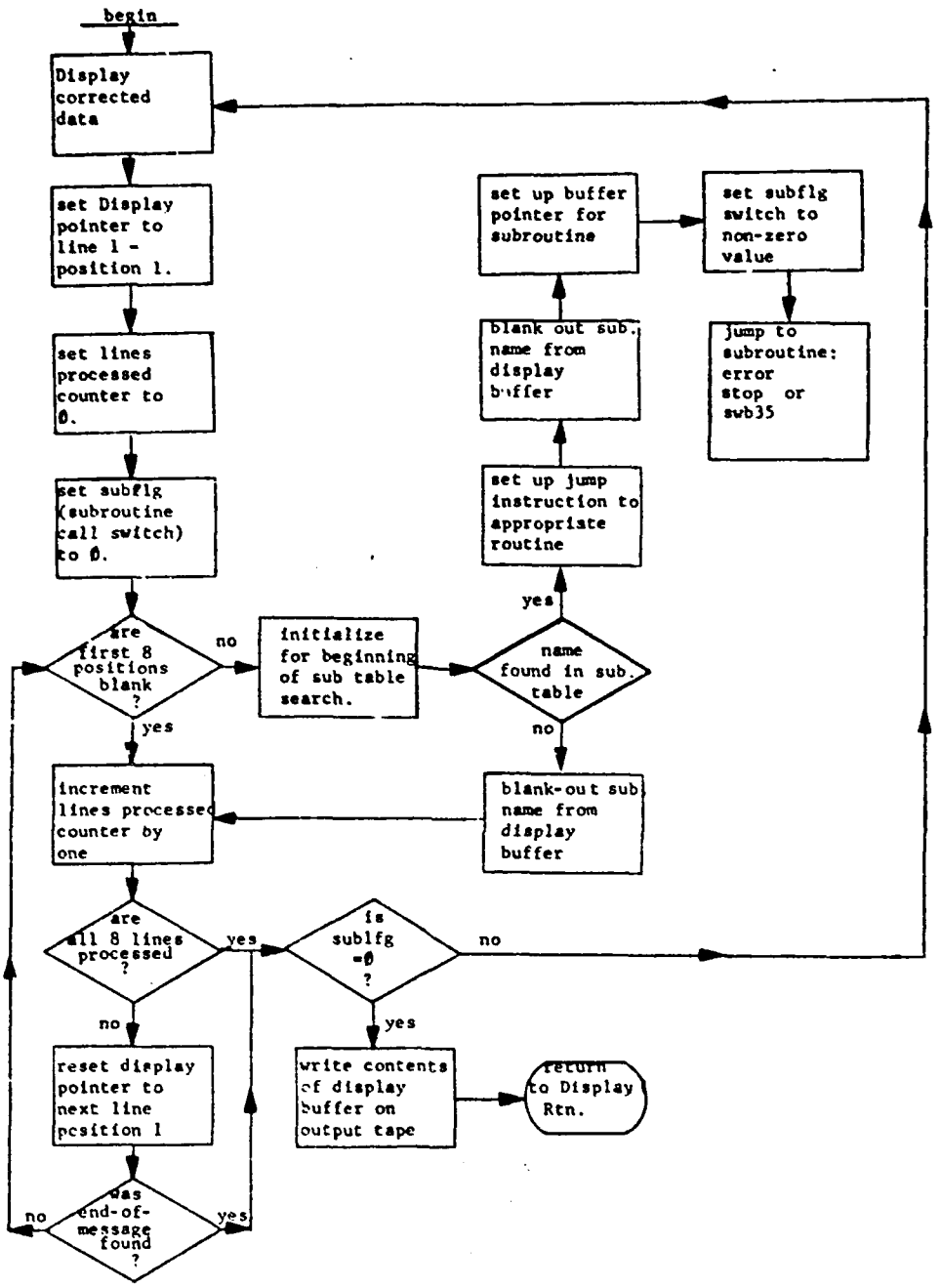


Fig. 4 - 11 Flow diagram for Manual Data Edit routine

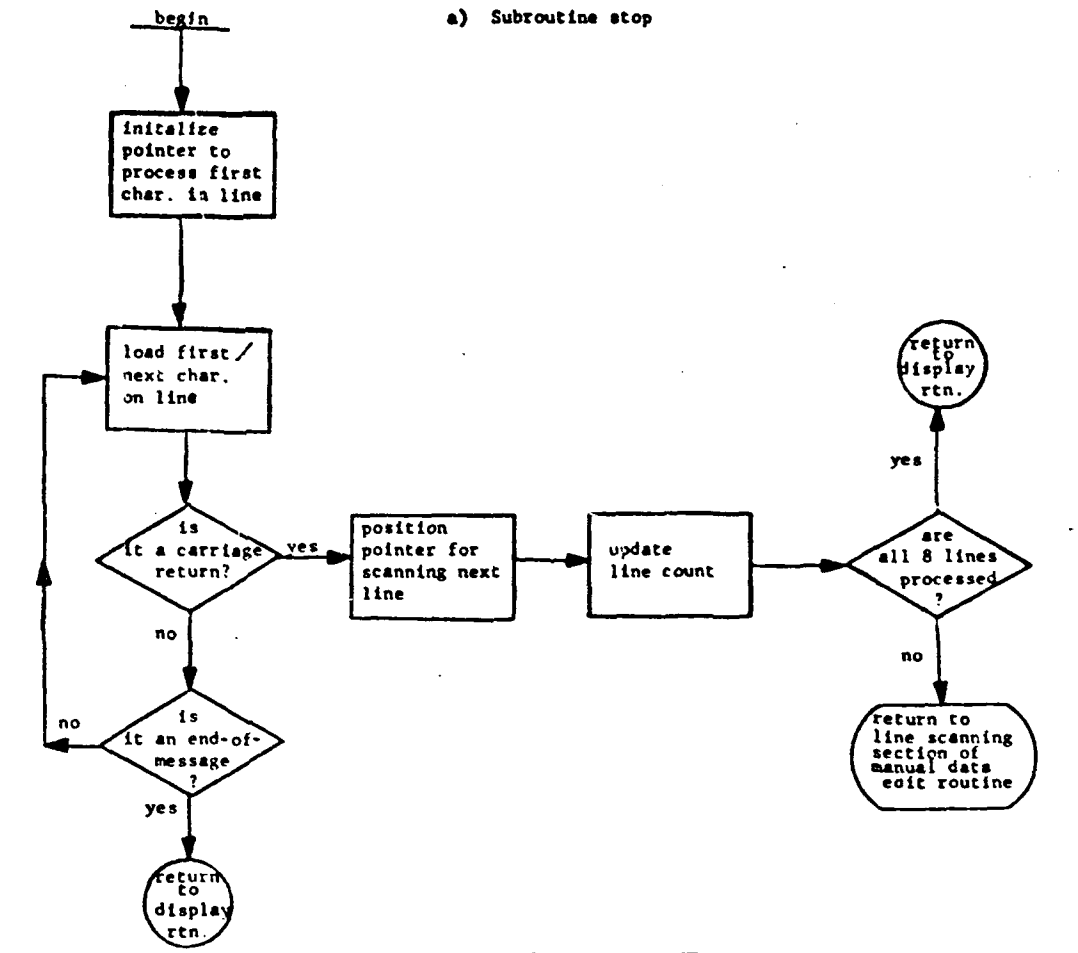


Fig. 4 - 11 (cont'd)

c) Subroutine SWB35

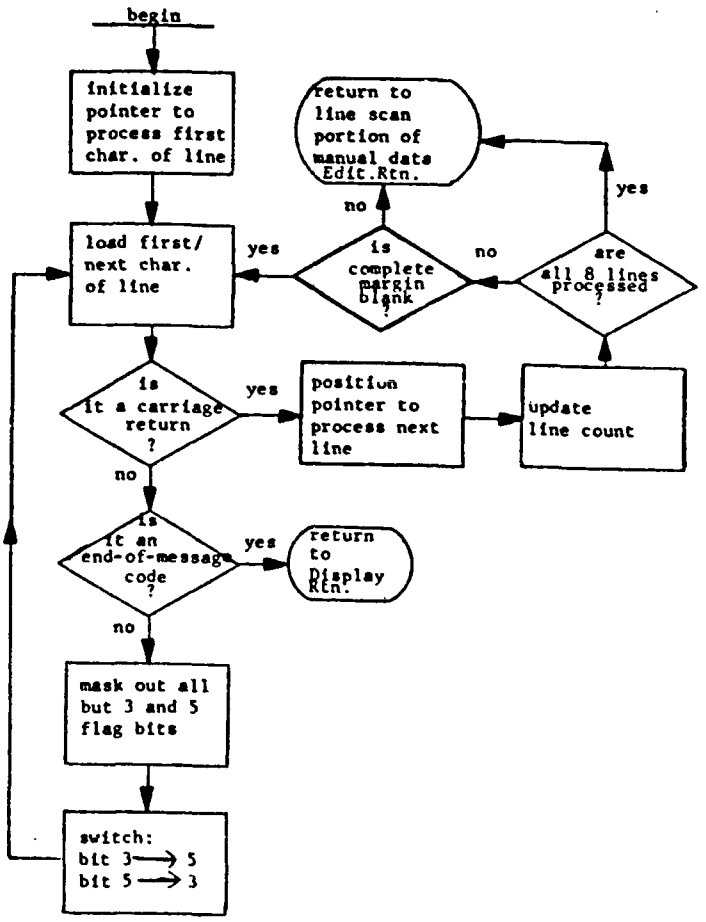


Fig. 4 - 11 (cont'd)

After an operator has selected the editing subroutine(s) which he intends to use in correcting the text, control is returned to the MANUAL DATA EDIT routine. This routine will interpret the operator's inserted marks and call the appropriate subroutine(s). The processed text will then be redisplayed on the upper half of the Datacom scope, as shown in Fig. 4-12b.

In Fig. 4-12a a different type font is used to represent color shift characters, e.g., ECP and DISPLAY in the first lines.

4.12 CODE CONVERSION (CONVRT) ROUTINE

4.12.1 Purpose

The purpose of the CODE CONVERSION routine (see Fig. 4-13) is to convert the corrected text from Universal to Print Reader code.

4.12.2 Input

The input for this routine is the magnetic output tape of the DISPLAY routine.

4.12.3 Description

The CODE CONVERSION routine reads and processes a single input block (512 words) at one time. Each Universal code is examined to determine if it is part of a color series. Those codes which are not part of a color series are converted using a table lookup technique, and the corresponding Print Reader code is stored in the output buffer. Each time a character is stored in the output buffer a check is made to determine if the buffer is full. When the buffer is full it is punched onto paper tape.

If the Universal code indicates that this character is part of a color series, it must then be determined if it is the first or last character in the series and set the appropriate flags. Only the first and last characters of a color series are accompanied by a color shift code in addition to the corresponding Print Reader code. The first character of a color series is preceded by a color shift code and the last character of a color series is followed by a color shift code.

The end of the input data is signaled by a double end-of-file block. Single end-of-file blocks are ignored by this routine and the next block in the input tape sequence is read.

4.12.4 Output

The output of the CODE CONVERSION routine, the final routine of the Spelling Correction program, consists of a punched paper tape in Print Reader codes. This tape can then be listed to obtain a hard copy of the original Print Reader input text with corrections inserted.

4.13 GENERATION OF SIMULATED DATA (GENDAT) PROGRAM

4.13.1 Purpose

SWB35	Issue a command to ECP to DISPLAY data as described
Stop	under DSPLY above,
SWB35	and then to accept data from the console device. ECP
	first outputs the
	segment designated in the operand field until either a
	STOP CODE is encountered
	or 512 WORDS have been displayed.
	ECP then waits until the programmer completes his

(a) Lower half of display

	Issue a command to ECP to display data as described
	under DSPLY above,
	and then to accept data from the CONSOLE DEVICE ECP
	first outputs the
	segment designated in the operand field until either a
	stop code is encountered
	or 512 words have been DISPLAYED.
	ECP then waits until the programmer completes his

(b) Upper half of display

Fig. 4-12 Examples of 408-A Datacom Display when using MANUAL DATA EDIT routine

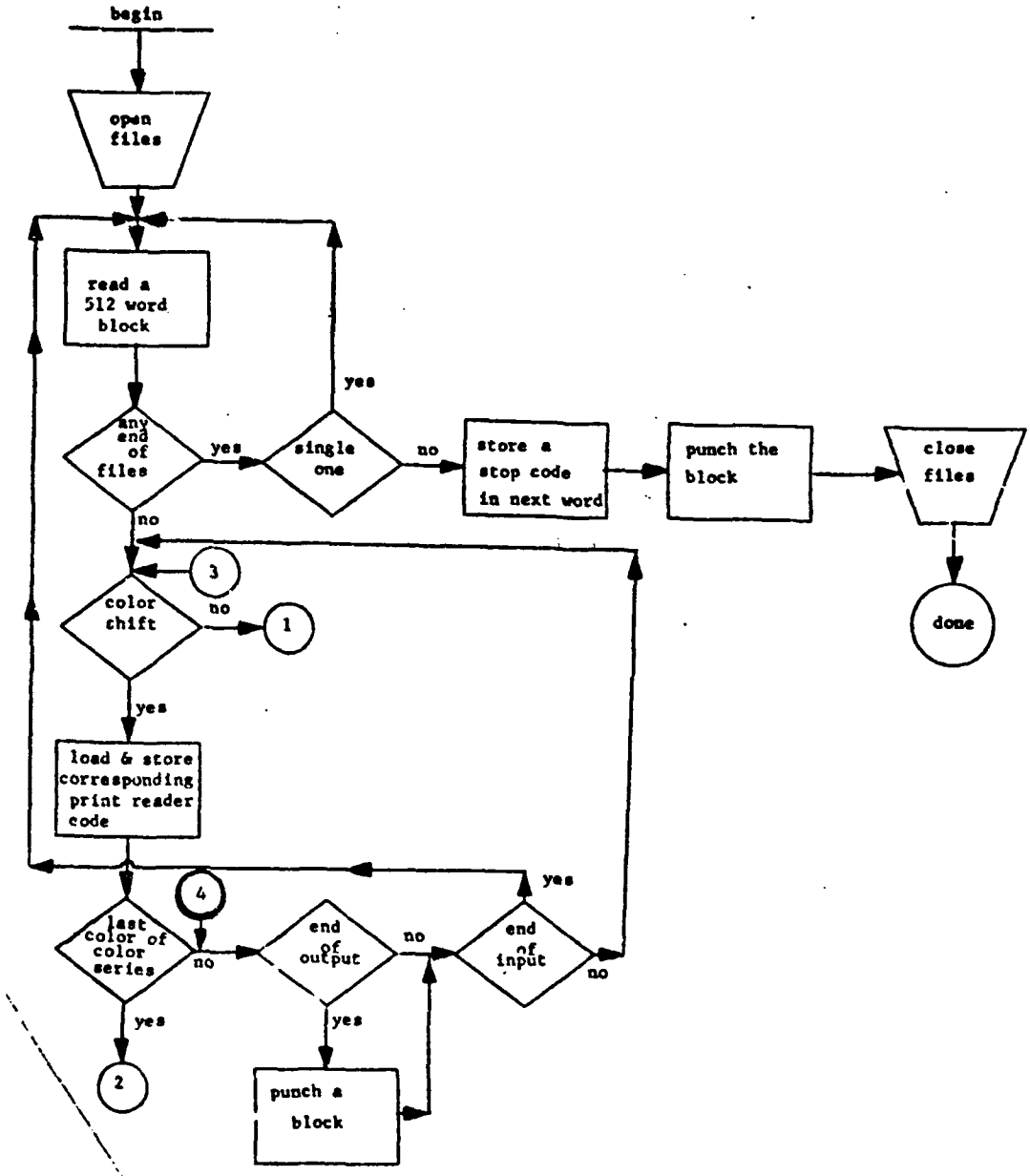


Fig. 4 - 13 Flow diagram for Code Conversion routine

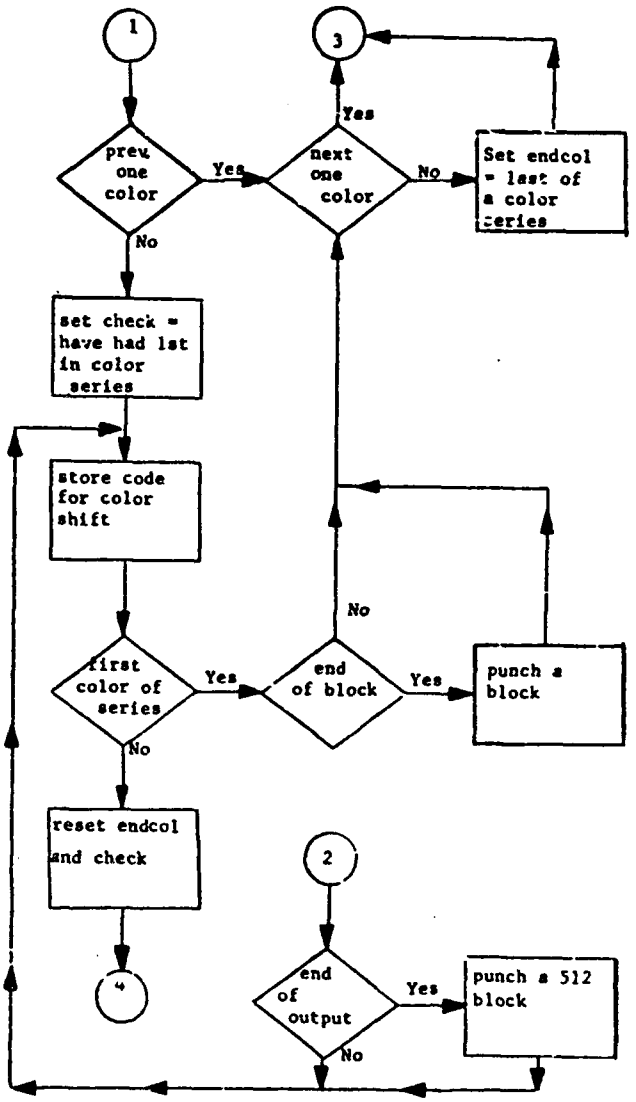


Fig. 4 - 13 (cont'd)

The purpose of the GENDAT program (see Fig. 4-14) is to create data which can be used as input in debugging the Spelling Correction routines. This proved necessary because at the time the routines were being debugged it was not possible to run the SRC routine. All but four of the routines, PEELER, SORT, MERGE, and SHORTWORD use the output of the SRC in one way or another (i.e., either directly using the output just as it came from this routine, or indirectly using the output of a routine which used the SRC's output).

4.13.2 Input

The input for this program is created by the user. It is done by typing in the data on the user's typewriter, after the message "okay type in data" has been typed out by the program.

4.13.3 Description

One generates data in blocks of 512_{10} computer words by using the RADCAP KEY IN instruction. A data word is typed just as it is to appear, except that at the end of each data word (which can be a space, a mark of punctuation, an actual word, etc.) a carriage return is placed. This is done so that the program will know when a data word is complete. At the end of the last data word a left parenthesis is typed; thus, neither a carriage return nor a left parenthesis can be used in a data word. If it is necessary to use either of these characters as a data word, the marks for end-of-word and end-of-data can be changed to any keyboard character that is not used as a data word.

Before the message to type in data has appeared, two leader words are written in the output buffer. They are used only by the SORT routine, so if data is not being created for this routine, they should not be written (see Section 4.13.5). Typing a left parenthesis returns control to the GENDAT program which then displays the block typed in by the user. Unused locations are automatically zeroed out.

Except for the first and last routines of the Spelling Correction program, all routines accept six additional words (descriptive words) for each data word. Thus, the GENDAT program must also generate these additional words. Of the six descriptive words, two of them, the begin and end words (174_8 and 175_8 respectively) remain constant. Two other words, document number and type code, are set and remain constant for all data words which are typed in. Only two words change their value during the execution of the GENDAT program: the character count and item position within the document.

As a data word is being stored by the GENDAT program, its character count is computed. When an end-of-word mark is encountered, the item position counter is increased by one and the word and its six descriptive words are stored in the output buffer; then the next data word is dealt with. When an end-of-data mark is encountered, the output buffer is written on the output tape five times and the program terminates.

4.13.4 Output

The output tape generated by this routine is identical in format to actual data. All unused locations at the end of the blocks contain zeros.

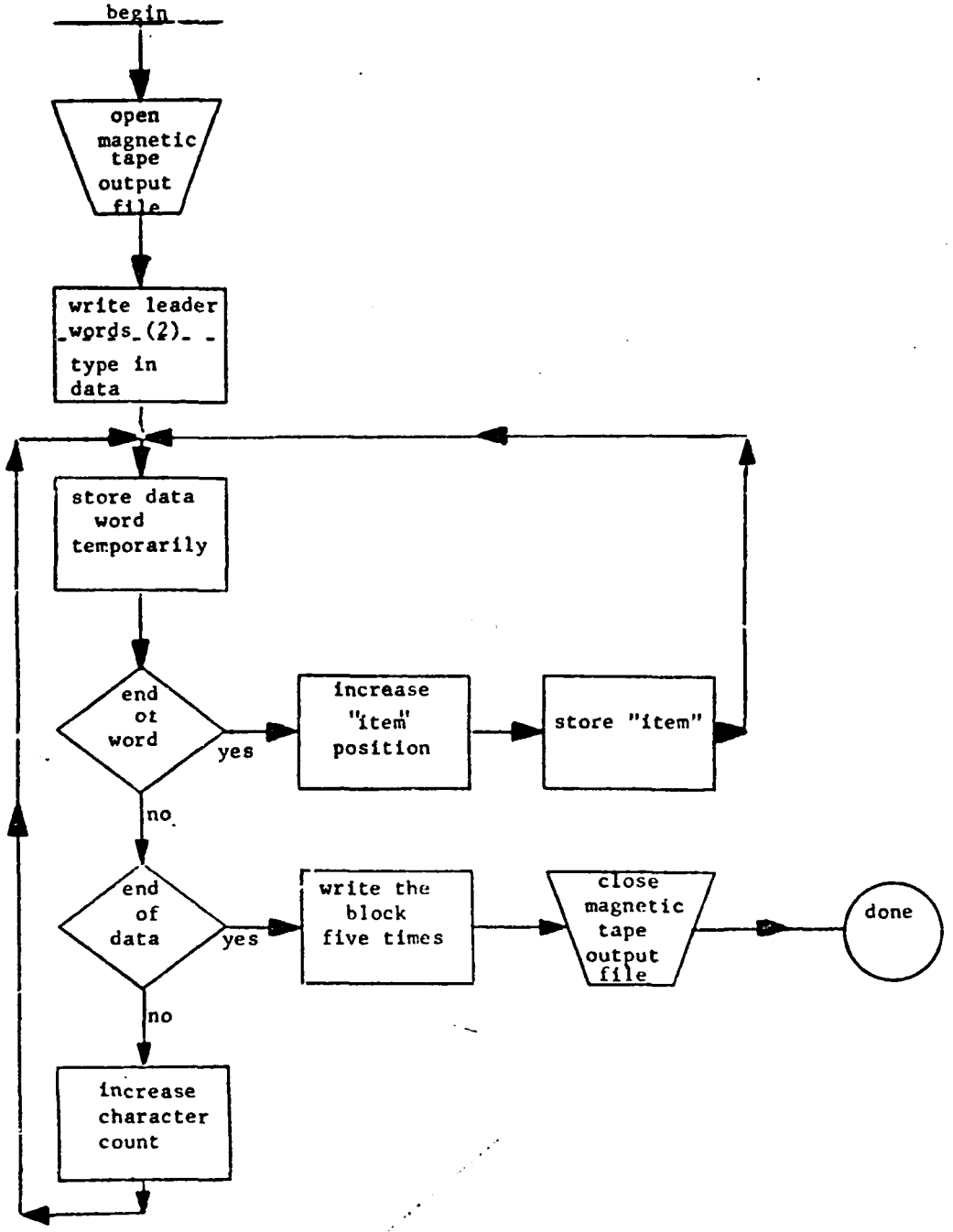


Fig. 4 - 14 Flow diagram for GENDAT routine

4.13.5 Changes

The following changes can be made, when desired, in the program.

1. The end-of-word mark can be changed from a carriage return to any character that appears on the typewriter keyboard simply by changing what is stored in location "cret" to the value for the new end-of-word mark. Note, however, that whatever is used as the end-of-word mark can be used only for that purpose and cannot be used as a data word or as part of a data word.

2. The end-of-data mark can be changed from a left parenthesis in the same way that the end-of-word mark was changed, i.e., by changing the value in location "endat" to that of some other keyboard character and using it uniquely for that purpose in the program.

3. For all routines except SORT, the two words written at the beginning of each block are not needed; to make data for these routines it is necessary to remove the cards which store these two words and to set the initial increment of EE, the location where the complete item is stored, to zero.

4. In the RE-SORT routine, it is essential that the item position of the words not be in ascending order. The reason for this is that the purpose of the program is to accomplish this very thing. Thus, initially the location "itemct" is set to some other number, X, and instead of adding one to the cell "itemct" every time a complete data word is found, one is subtracted from it at that point in the program.

5. The type code may also be changed merely by changing the contents of the location "type." A list of what the different codes mean is included in Appendix A.

6. If more than one block of data is desired, it can be obtained by setting a counter at the start of the program and increasing it every time a block is written until some maximum volume has been reached. After the blocks have been written, the program will zero out the output buffer and wait for more data to be typed.

5. OPERATING PROCEDURES*

5.1 PEELER OPERATION

1. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

2. On the user's typewriter, call up job 001.

3. After the routine is loaded, a message to assign file 001 is typed out on the SCO typewriter. Assign the output tape of RADC's Printer Code Conversion program.

4. Next, a message to assign a scratch file will be typed; assign any tape. This tape is used as the output tape.

5. During the execution of the program, as each block of input is read, it is displayed on the typewriter and when all blocks have been read, the letter "e" is displayed.

6. The routine is finished when the message on the user's console (i.e., the typewriter) states that job 001 has terminated. At the same time, a message to remove the two files is written on the SCO typewriter.

7. File 001, the input, is not used again. File 002, the output, is used by the SORT routine.

5.2 SORT OPERATION

1. This routine is also referred to as SOALNO.

2. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

3. On the user's typewriter, call up job 014.

4. When the message to assign file 001 is typed out, assign the output tape of the PEELER routine.

* All job numbers and file numbers for the operating instructions are given in octal code.

5. When the message to assign a scratch file appears, assign any tape. This tape is used as the output tape.

6. After both files have been assigned, a message saying "type batch number in octal" is written on the user's typewriter. Type in the number of documents that are to be batched together.

7. When the routine is finished, a message stating job 014 has terminated will appear on the user's console. A message to remove the two files will be typed on the SCO typewriter.

8. File 001, the input, is not used again. File 002, the output, is used by the MERGE routine.

5.3 MERGE OPERATION

1. Mount the program tape on the tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

2. On the user's typewriter, call up job 750.

3. When the job is loaded, a message to assign a scratch tape will appear. This tape is used as the final output tape.

4. After a small portion of the routine has been executed, a message to assign file 043 is typed out. This is the input file and is the same file outputted by the SORT routine.

5. Two other files are used, but they are disk files and their assignments are automatic.

6. When the routine is finished, a message stating job 750 has terminated is typed out on the user's console. A message to remove the two files is written on the SCO typewriter.

7. File 043, the input, is not used again. File 046, the output, is used by the SHORTWORD routine.

5.4 SHORTWORD OPERATION

1. This routine is also referred to as SHRTWD.

2. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

3. On the user's typewriter, call up job 225.

4. When the message to assign file 025 is typed out, assign the output tape of the MERGE routine.

5. When the message to assign a scratch file appears, assign any tape. This tape is used as the output tape.

6. While the routine is being executed, no messages are written and no assignments are made.

7. The routine is finished when the message on the user's console states that job 225 has terminated and the SCO typewriter requests the two files be removed.

8. File 025, the input, is not used again. File 026, the output, is used by the SRC routine.

5.5 LOAD DISK OPERATION

1. This routine is also referred to as LODDSK.

2. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it had already been assigned.

3. Load the paper tape reader with a dictionary tape.

4. Call up job number 507.

5. When the job is loaded, a message will appear to assign file 033. At this point, assign the paper tape reader.

6. When the job is finished, one of two messages appears on the SCO typewriter:

   a. DISK LOADED WITH FULL DICTIONARY.
   b. DISK FULL PREMATURELY, CHAR LENGTH...
      (a number will follow CHAR LENGTH indicating the last character length loaded).

7. This program will be used again to load more dictionaries or the remainder of dictionaries.

5.6 SHIFT REGISTER COMPARATOR OPERATION

1. This routine is also referred to as SRC.

2. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

3. Mount the output tape from the SHORTWORD routine on a tape drive.

4. Mount a scratch tape on another tape drive.

5. Call up job 510.

6. When the job is loaded, a message will appear to assign file 035. This is the input tape which was the final output tape of the SHORTWORD routine.

7. Another message will appear requesting a scratch tape for file 036. This will be the output tape.

8. When the job is finished, a message to remove the two files is written on the SCO typewriter.

9. File 035, the input file, is not used again. File 036, the output file, is used as input to the RE-SORT routine.

5.7 RE-SORT OPERATION

1. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

2. On the user's typewriter call up job 004.

3. When the message to assign file 001 is typed out, assign the output tape of the SRC routine.

4. When a message to assign a scratch tape appears, assign any tape. This will be used as the final output tape.

5. Two other files are used, but they are disk files and their assignments are automatic.

6. When the routine is finished, a message stating that job 004 has terminated is typed out on the user's console. A message to remove the file will appear on the SCO typewriter.

7. File 001, the input, is not used again. File 004, the output, is used by the DISPLAY routine and the two statistics routines.

5.8 STATISTICS I OPERATION

1. This routine is also known as WORST.

2. It is not necessary to run this routine next. It can be run at any point once the RE-SORT output has been obtained.

3. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

4. On the user's typewriter, call up job 751.

5. When the message to assign file 002 is typed out, assign the output tape of the RE-SORT routine.

6. During the execution of the routine, the output will be displayed on the user's typewriter.

7. When the message that job 751 has terminated appears on the user's console, the routine is finished. A message to remove the file will appear on the SCO typewriter.

8. File 002, the input, is used again unless PONCW and DISPLAY have already been executed. The typewriter display is the output for this routine.

5.9 STATISTICS II OPERATION

1. This routine is also known as PONCW.

2. It is not necessary to run this routine next. It can be run at any point once the RE-SORT output has been obtained.

3. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

4. Make sure that unit 7, the printer, is free.

5. On the user's typewriter call up job 250.

6. When the message to assign file 600 is typed out, assign the output tape of the RE-SORT routine.

7. During the execution of the routine, and when the printer is to be used, a message to assign unit 7 appears on the SCO typewriter.

8. When the routine is finished, a message stating that job 250 has terminated is typed out on the user's console. A message to remove the file appears on the SCO typewriter.

9. File 600, the input, is used again unless both DISPLAY and WORST have been executed. The output is the printing which appeared during the execution of the routine.

5.10 DISPLAY AND MANUAL DATA EDIT OPERATION

1. The MANUAL DATA EDIT routine is also referred to as MDE.

2. It is not necessary to run this routine next. It can be run anytime after the RE-SORT but before the CODE CONVERSION routine.

3. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

4. On the Datacom, call up job 251.

5. When the message to assign file 602 appears, assign the output tape of the RE-SORT ROUTINE.

6. When a message to assign a scratch tape appears, assign any tape. This is the output tape.

7. During the execution of the routine and before a block is written on tape, it is displayed on the upper half of the Datacom scope. At this point it is possible to manually correct the text. See description of MANUAL DATA EDIT routine for instructions on correcting the text. After the text is corrected, the READOUT button on the Datacom console is pushed to return control to the Spelling Correction routines.

8. The routine is finished when the message on the Datacom states that job 251 has terminated. Just before termination a message will appear on the SCO typewriter to remove the two files.

9. File 602, the input, is used again, unless both PONCW and WORST have been executed. File 603, the output, is used by the CODE CONVERSION routine.

5.11 CODE CONVERSION OPERATION

1. This routine is also called CONVRT.

2. It is not necessary to run this routine next. It can be run any time after the DISPLAY routine.

3. Mount the program tape on a tape unit and assign that unit as the system's tape unit, unless it has already been assigned.

4. Make sure unit 2, the punch, is free.

5. Call up job 275 on the user's typewriter.

6. When the message to assign file 610 is typed, assign it the output of the DISPLAY routine.

7. When the message to assign unit 2 is typed, assign the punch.

8. When the routine is finished, a message stating that job 275 has terminated is typed out on the user's console. Prior to termination a message will appear on the SCO typewriter to remove the two files.

9. File 610, the input, is not used again. File 611, the output, is used to obtain a hard copy of the text.

5.12 GENDAT OPERATION

1. Mount the program tape on a tape drive and assign that unit as the system's tape unit, unless it has already been assigned.

2. On the user's typewriter, call up job 013.

3. When the message to assign a scratch tape is typed out, assign any free tape.

4. When it is time for the input data to be typed in, the message "okay type in data" is typed out on the user's typewriter. The data is entered on this console according to the format stated in the description of GENDAT.

5. When the program is finished, a message on the user's console states that job 013 has terminated. On the SCO typewriter there is a message to remove the file.

6. File 001, the output, is used as input to one of the Spelling Correction routines.

Appendix A

DESCRIPTION OF AN "ITEM"

An "item" is composed of a data word and six descriptive words and appears as follows:

Word	Description
1	character count
2	type code
3	document number
4	item position within document
5	begin word code
.	
.	data word
.	
n	end word code

An item is of variable length n, where $7 \leq n \leq 96$ computer words. The data word is the original document information. The six descriptive words are appended to the data word in the PEELER routine to assist in data handling and correction procedures. A data word may be a single space, a mark of punctuation, a completely alphabetic stream of characters, numbers, etc. In any case, the first character of a data word always begins in the sixth word of the item.

Following is an explanation of the six descriptive words which make up an item:

1. Character Count — Indicates the number of characters in the data word.

2. Type Code — Indicates the composition of the data word. It may indicate that the word is pure alphabetic, numeric, alphabetic with confusion character, alphabetic with

best guess, etc. The code format is:

x	x	x	x	y	y	y	y	y	y	y	y

Where xxxx being indicates

0000	alphabetic characters only
0100	alphabetic characters with confusion characters
0010	alphabetic characters with best guess
0001	alphabetic characters with hyphen
0110	alphabetic characters with best guess and confusion characters
1000	numeric characters
1100	anything else

The y portion of the word contains a count, right adjusted, for the number of confusion or best guess occurrences in the data word.

For a few special characters, all 12 bits of the type code word are used. They are:

-1 - - -	punctuation
-2 - - -	hyphen
-3 - - -	start of document

3. Document Number — Indicates which document the data word belongs to. The numbering is initialized at one and is incremented by one every time a start document code is encountered.

4. Item Position — Indicates the relative position of the data word within the original document. The numbering starts at one and increases by one for each data word. The numbering restarts each time another document is encountered.

5. Begin Word — Code word, 174_8 required by the shift register comparator. The code is right adjusted in the computer word leaving the five leftmost bits as indicators for the SRC and SHORTWORD routines. A corrected data word will have a zero in the leftmost bit and a number representing the tolerance level required for correcting this word in the remaining four bits. An uncorrected data word will have a one in the leftmost bit and a number representing the highest tolerance level used in attempting to correct the data word in the remaining four bits. The words which were not processed by the SRC or SHORTWORD routines have only the 174_8 code in the computer word.

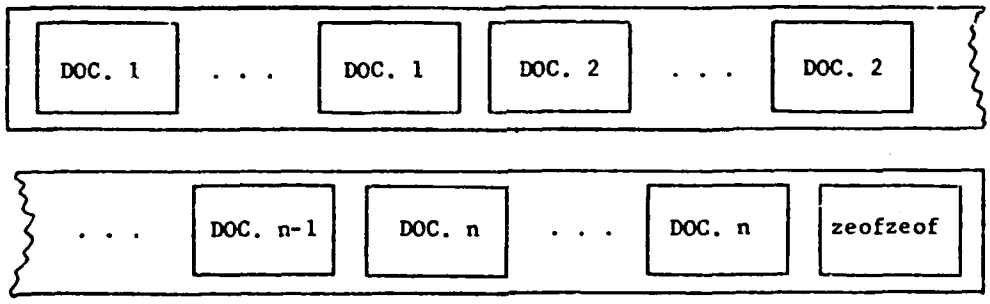
6. End Word — Code word, 175_8, required by the shift register comparator. It is always in the (6+n) word position of the item, where n is the character count of the data word.

Appendix B

BATCHING SCHEME USED IN SPELLING CORRECTION PROGRAM

The initial input to the Spelling Correction SORT routine is a magnetic tape consisting of blocks of 512 computer words. The first word of each block contains a document number indicating the document to which the data words in the block belong. A block of information contains only words from a single document. Unused words in a block, if any, are set to zero. The last data block on the magnetic tape is followed by a double end-of-file block (see Appendix C for a description of end-of-file blocks).

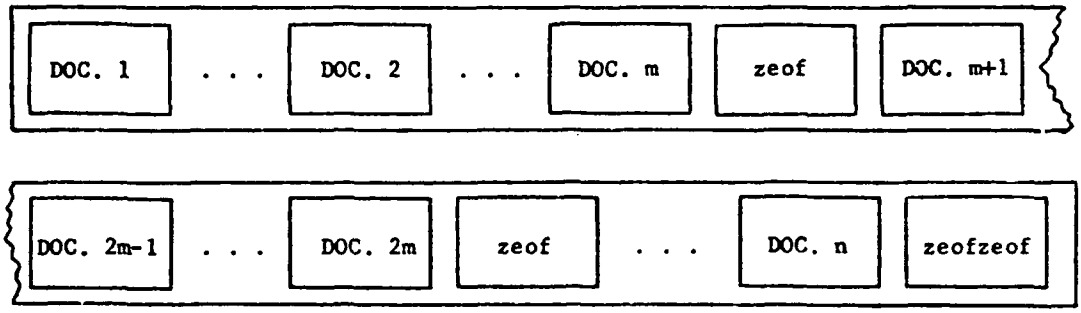
An example of the data configuration of n documents for input to the SORT routine is as follows:



To increase the data handling efficiency in both sorting and merging operations and in the dictionary lookup techniques, a batching capability has been added to the SORT routine.

When the SORT routine is initialized, a request will appear on the ECP Selectric typewriter for the operator to enter an integer indicating the number of documents to be batched. The above mentioned operations will then be followed by a single end-of-file block. The last batch will be followed by a double end-of-file block.

Following is an example of the output tape data configuration from the SORT routine where the n documents have been batched m documents to a batch:



The batching configuration is retained throughout the remaining Spelling Correction operations until the last routine in the sequence, CODE CONVERT, restores the corrected data to its original codes and format.

Appendix C

END-OF-FILE BLOCKS

The single end-of-file block used in the Spelling Correction routines is either automatically generated by using a RADCAP CLSOP instruction or program generated by placing the proper bit configuration in a reserved storage block and writing this block on a magnetic tape.

Either method will generate the single end-of-file block shown in the following diagram:

Single End-of-File Block

Position	Octal	Universal
Word 1	0077	z
Word 2	0025	e
Word 3	0051	o
Word 4	0027	f
Word 5	0012	Blank
o	o	o
o	o	o
o	o	o
Word 512	0012	Blank

The double end-of-file block used in the Spelling Correction routines can only be program generated. The following diagram shows an example of the bit configuration:

Double End-of-File Block

Position	Octal	Universal
Word 1	0077	z
Word 2	0025	e
Word 3	0051	o
Word 4	0027	f
Word 5	0077	z
Word 6	0025	e
Word 7	0051	o
Word 8	0027	f
Word 9	0012	Blank
o	o	o
o	o	o
o	o	o
Word 512	0012	Blank

The end-of-file indicator routine used in the Spelling Correction program does not require the contents of words 5 through 512 in the single end-of-file block or words 9 through 512 in the double end-of-file block to be blanks.

Appendix D

SEQUENCE FOR SETTING SHIFT REGISTER COMPARATOR CONDITIONS

This operation sets the control flip-flops for the compare network between SR1 and SR2:

7100	xx = threshold value (0 indicates the exact match mode)	
0004		
3700	a - allow "confusion"	) 0 indicates
0201	b - allow "don't care"	) reset for:
00xx	c - inhibit transfer of confusion bit	) 1 indicates
abcd	d - prefix mode	) set for
zzzz	zzzz = 0 indicates operation performed, ≠ 0 indicates SRC busy	

(See SRC hardware manual for further details on condition settings.)

Appendix E

EXAMPLES OF SHIFT REGISTER COMPARATOR ROUTINE FLAG BITS

Based on the conditions of a 12_{10} character data word that was not corrected and the threshold level was lowered to 6_{10}, the bit configuration of word 5 would be as follows:

Before lookup

WORD	DESCRIPTION	EXAMPLE	BIT CONFIGURATION
			11 10 9 8 7 6 5 4 3 2 1 0
5	begin word	174_8	0 0 0 0 0 1 1 1 1 1 0 0

After lookup (not corrected)

5	flags and begin word	5574	1 0 1 1 0 1 1 1 1 1 0 0

If the same conditions existed and the data word was considered corrected, word 5 would be as follows:

After lookup (corrected)

5	flags and begin word	1574	0 0 1 1 0 1 1 1 1 1 0 0

Appendix F

EXAMPLE OF OUTPUT OF STATISTICS I

Batch 1

Character Length	Frequency	Best Guess	Confusion	Confusion Plus Best Guess	Corrected	Uncorrected
1	0	0	0	0	0	0
2	250	40	11	15	235	15
3	473	93	34	27	446	27
4	379	56	25	18	350	29
5	400	47	10	31	367	33
6	228	12	12	0	208	20
7	212	18	24	6	199	13
8	103	25	19	8	96	7
9	27	4	2	0	25	2
10	10	2	1	1	10	0
11	12	4	2	0	9	3
12	9	3	2	1	8	1
13	6	2	1	0	5	1
14	3	1	0	0	3	0
15	0	0	0	0	0	0
16	2	0	0	0	2	0
17	1	1	0	0	1	0
18	1	0	1	0	0	1

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Itek Corporation Lexington, Massachusetts	2a REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b GROUP

3 REPORT TITLE
Computer Program for Automatic Spelling Correction

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report

5 AUTHOR(S) (Last name, first name, initial)
O'Brien, Joseph A.

6 REPORT DATE	7a TOTAL NO. OF PAGES	7b NO. OF REFS
March 1967	108	

8a CONTRACT OR GRANT NO.	9a ORIGINATOR'S REPORT NUMBER(S)
AF30(602)-3484 b PROJECT NO 4594 c d	66-8428-1 9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report) RADC-TR-66-696

10 AVAILABILITY/LIMITATION NOTICES
This document is subject to special export controls and each transmittal to foreign governments, foreign nationals or representatives thereto may be made only with prior approval of RADC (EMLI), GAFB, NY 13440.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
Louis Comito Project Engineer (EMIIF)	Rome Air Development Center (EMIIF) Griffiss AFB, NY 13440

13 ABSTRACT

This technical documentary report, prepared under Contract AF30(602)-3484, describes the logic and operation of the Spelling Correction Program prepared by Itek Corporation for RADC.

The program is written in RADCAP language to operate on a CDC 8090 Computer under control of the RADC Experimental Computer Complex (ECC).

DD FORM 1 JAN 64 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Programming Computers						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification